

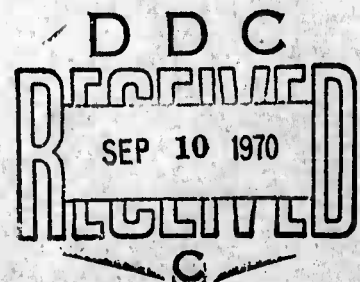
AD 711342

August 1970

ADVANCED RESEARCH PROJECTS AGENCY
SEMIANNUAL TECHNICAL REPORT

G. Estrin
L. Kleinrock
M. Melkanoff
R. R. Muntz

Reproduced by the
CLEARINGHOUSE
for Federal Scientific & Technical
Information Springfield Va. 22151



This document has been approved
for public release and sale; its
distribution is unlimited.

123

**BEST
AVAILABLE COPY**

ADVANCED RESEARCH PROJECTS AGENCY

SEMIANNUAL TECHNICAL REPORT

August 15, 1970

COMPUTER NETWORK RESEARCH

DAHC-15-69-C-0285

Computer Science Department
School of Engineering and Applied Science
University of California, Los Angeles

Principal Investigator: Leonard Kleinrock

Co-Principal Investigators: Gerald Estrin
Michel Melkanoff
Richard R. Muntz

UNCLASSIFIED

ADVANCED RESEARCH PROJECTS AGENCY

SEMIANNUAL TECHNICAL REPORT

August 15, 1970

Project Computer Network Research ARPA Order Number 1380

Contract Number DAHC-15-69-C-0285

Effective Date of Contract 4/1/69 Contract Expiration Date 10/31/70

Amount of Contract \$873,109*

Contractor: School of Engineering and Applied Science
Computer Science Department
405 Hilgard Avenue
University of California
Los Angeles, California 90024

Project Scientist and Principal Investigator Dr. Leonard Kleinrock

Phone (213) 825-2543

*Since this contract has had a number of amendments since its initial acceptance, we list below the pertinent chronology:

<u>Date Funds Requested</u>	<u>Period Covered</u>	<u>Amount</u>	<u>Status</u>
November 1968	4/1/69-10/31/69	\$229,300	Approved
November 1968	11/1/69-10/31/70	\$344,413	"
August 1969	4/1/69-10/31/70	\$ 69,396	"
August 1969	11/1/70-6/30/71	\$243,345	In process
December 1969	12/1/69-6/30/70	\$230,000	Approved
December 1969	7/1/70-6/30/71	\$300,000	In process

TABLE OF CONTENTS

<u>Section</u>	<u>Page</u>
1. INTRODUCTION	1
2. MATHEMATICAL MODELLING AND ANALYSIS OF COMPUTER SYSTEMS	2
3. NETWORK MEASUREMENTS	8
4. NETWORK AND SYSTEMS SOFTWARE	17
5. CONCLUSIONS AND SELF-EVALUATION	20
REFERENCES	21
APPENDICES	24
APPENDIX A	
Swap-Time Considerations in Time-Shared Systems, by Leonard Kleinrock	25
APPENDIX B	
A Continuum of Time-Sharing Scheduling Algorithms, by Leonard Kleinrock	33
APPENDIX C	
Multilevel Processor-Sharing Queueing Models for Time-Shared Systems, by L. Kleinrock and R. R. Muntz	40
APPENDIX D	
Buffer Behavior for Batch Poisson Arrivals and Single Constant Output, by Wesley W. Chu	49

	<u>Page</u>
APPENDIX E	
Selection of Optimal Transmission Rate for Statistical Multiplexors, by Wesley W. Chu	73
APPENDIX F	
Computer Network Studies, by Gary Fultz	78
APPENDIX G	
Dynamic Storage Allocation for Binary Search Trees in a Two-Level Memory, by Richard Muntz and Robert Uzgalis	99

ADVANCED RESEARCH PROJECTS AGENCY

SEMIANNUAL TECHNICAL REPORT

August 15, 1970

1. INTRODUCTION

~~The goal of this project~~ ^{THE REPORT} is to create an environment suitable for high quality computer research activities in the understanding and in the development of methods for information processing. One principle area of activity is in the mathematical modelling of computer systems. This includes modelling time-shared systems, memory hierarchical systems, and the ARPA experimental computer network. Another principle area is our network measurement procedures which we intend to use in studying the network behavior as well as for evaluating the validity of our modelling and analytical work. Lastly, we are responsible for establishing a suitable HOST-HOST software protocol for use in the network.) A

Progress toward the goals listed above has progressed well since the initiation of this contract. Some of our earlier work was reported upon in our Semiannual Technical Report dated February 15, 1970. References [1] through [6] listed at the end of this Technical Report include work which took place prior to the current reporting period.

In the following three sections we report upon the progress which has taken place in this current reporting period. Section 2 describes some of the mathematical modelling and analytical work for computer systems which we have developed. The ARPA research group at UCLA has established itself

as perhaps the strongest analytical group for computer systems modelling in the country. In Section 2.1 we describe our work in the analysis of time-shared computer systems. In Section 2.2 we give results for computer-communication network investigations. Finally, in Section 2.3 we describe a recent paper on storage allocation in hierarchical memories. Section 3 is a report on our measurement activity in the ARPA Network. Progress here has been strong but it is not as yet highly visible outside of UCLA. In the next six months we intend to carry out elaborate measurement experiments which will involve interaction with other nodes in the network and will permit measurement work to take place at remote locations with the aid of the UCLA system. Section 4 describes progress in the area of software; in particular, progress on the network protocol and the growth of our time-sharing system at UCLA are described. Section 5 offers some conclusions regarding the impact of the studies reported herein.

2. MATHEMATICAL MODELLING AND ANALYSIS OF COMPUTER SYSTEMS

Our effort in the modelling of computer systems has applied mainly to the following three areas: time-shared scheduling algorithms; computer-communication networks; and storage allocation in memory hierarchies.

2.1. Time-Shared Systems Analysis

This period has been most profitable in generating new results in the field of time-shared scheduling analysis. A paper entitled "Swap Time Considerations in Time-Shared Systems" was published by Kleinrock in June 1970 [13]. In this paper a procedure was developed for calculating the average swap time expended on all those customers still remaining in the

time-sharing system for a very general class scheduling algorithms. An interesting set of examples is given in that paper to which this result is applied and which permit one to gain considerable insight into the operation of these example algorithms. For further details, see a copy of the paper given in Appendix A.

In the 1970 Spring Joint Computer Conference, Kleinrock gave a paper entitled "A Continuum of Time-Sharing Scheduling Algorithms" [9]. This work reports upon a new class of scheduling algorithms which, on the one hand permits a continuum of algorithms to be defined which range smoothly from batch processing (first-come-first-served) to the well-known round-robin scheduling algorithm, and on the other hand permit this entire class of algorithms to be implemented in a surprisingly easy fashion. The principle result in this paper shows that for the entire range of scheduling algorithms analyzed, the average response time conditioned on a service time of t seconds is equal to a constant plus a linear function of t . Furthermore, all of these algorithms yield the same value of response time when the service time equals the average service time. This paper is reproduced as Appendix B.

Carl Hsu, a graduate student working under contract, has made progress in extending the work reported upon in Ref. [9]. In particular, he has successfully obtained the Laplace transform for the waiting time distribution for the continuum of scheduling algorithms described above. In addition, under the supervision of Kleinrock, he is developing extended models which permit the continuum of algorithms to range beyond the round-robin case all the way to the foreground-background case. This will then permit

one to implement a class of well-understood algorithms which pass from the case of no discrimination on the basis of service time (first-come-first-served) through the most common scheduling algorithm (round-robin) finally to the most discriminatory algorithm on the basis of service time (foreground-background).

Recently, Kleinrock and Muntz [14] have successfully defined an extremely broad class of scheduling algorithms which permit the designer a large number of new degrees of freedom. These multilevel queueing disciplines allow different scheduling algorithms to be effective during a job's progress through the time-shared system based upon the amount of attained service. The results of this work are to be reported upon in Munich, Germany, at the Sixth International Teletraffic Congress, September 1970. The details of that paper are reproduced in Appendix C. In this work, an integral equation was developed which defined the waiting time in the case where round-robin was the discipline used at some internal level within the system. Until recently that integral equation remained unsolved; however, since publication of this paper, Kleinrock, Muntz, and Rodemich have solved that integral equation for a rather wide class of service time disciplines, and these results are reported upon Ref. [16] (currently in preparation). Moreover, the application of that result in a variety of interesting situations is currently being prepared as Ref. [17].

As pointed out many times before by this research group, two outstanding problems remain unsolved in the area of time-shared scheduling algorithms. The first is the problem of considering a time-shared system as containing multiple resources for which competition takes place. For

example, the storage capacity, the central processing unit, the input-output devices, and the data channels connecting these devices together and to the users are all resources which undergo competitive demand. It is clear that the single resource analyses (for example, those reported upon above) have reached a point beyond which it is perhaps not profitable to go before we understand considerably more about the multiple resource case. The second outstanding problem is that involving the cost of delay and defining in a careful and meaningful way an appropriate criterion of performance for time-shared systems. Once this is done, then optimization of scheduling algorithms and other resource allocation algorithms can be carried out. Regarding the first of these problems, some progress has been made by our group in the area of analyzing one aspect of user behavior as it affects the system performance. Chu has studied the effect of the bursty nature of user input over character-oriented input-output devices. This work is reported upon in Ref. [18] and forms Appendix D of this report. This problem grows out of studies on the design of statistical multiplexors which are useful in data communications problems.

2.2. Computer-Communication Nets

Our effort in the analysis and optimization of computer-communication networks has continued and accelerated since the last reporting period. In this section we note results which have been obtained in the areas of multiplexing, routing, optimum channel capacity assignment, and certain topological studies.

At the 1969 Fall Joint Computer Conference, Chu presented a paper on "Asynchronous Time Division Multiplexing for Time-Sharing Computer Systems"

[6]. More recently, he has continued that study under our support and has published a paper entitled "Selection of Optimal Transmission Rate for Statistical Multiplexors" [12]. In this paper he considers the trade-off between the cost of transmission between computers and the cost of duplicating storage files at each. This paper forms Appendix E.

We have for some time now been involved with the design of the routing discipline in the ARPA Network and are continuing this interest. The problem of studying a variety of message routing strategies forces one to create suitable models of the nodes in that network, as well as a means for describing the message flow. Gary Fultz, another graduate student, has been actively investigating this area, and Appendix F is devoted to his progress in computer network studies. The results so far indicate that the models we have created correspond very well to the simulation results over many operating ranges and operating modes. Jack Ziegler, also a graduate student, is investigating the propagation of nodal storage blocking effects within the network. Some analytical results have been obtained in this area, and he currently is completing the details on a digital simulation based upon an epidemiology model of the way in which this blocking phenomenon propagates through a network.

Our previous Semiannual Technical Report included a copy of Ref. [10] by Kleinrock on "Analytic and Simulation Methods in Computer Network Design." That paper was presented and published in the 1970 Spring Joint Computer Conference Proceedings.

Recently, Professor Cantor in the Mathematics Department has been contributing to our network effort at UCLA. His interest lies in network flow

theory and mathematical programming techniques. We have been collaborating on some shortest route algorithms for our ARPA Network simulation program. Recently, he completed a paper on "Non-Blocking Switching Networks" [15]; this work provides tighter upper bounds on the number of switches necessary to connect one set of input lines to a second set of output lines in a non-blocking network. Mario Gerla and Luigi Fratta have recently joined us as graduate students and are currently investigating connectivity, survivability, and optimum topological design for computer networks. This effort is beginning to receive the proper attention that it should at a modelling center such as ours.

2.3. Memory Allocation in Hierarchical Memories

A recent study by Muntz and Uzgalis on "Dynamic Storage Allocation for Binary Search Trees in a Two-Level Memory" is included as Appendix G in this report (Ref. [18]). This paper presents a remarkably efficient algorithm for placing storage entries into a hierarchical memory in the case where the data set is tree-structured. An analysis has shown that the technique is near optimum.

3. NETWORK MEASUREMENTS

The network measurement activity has consisted of two major efforts, (1) the development of the network measurement and control programs which handle the measurement data and generate artificial traffic, and (2) the analytical studies involved with the data analysis and relating the measurements to the network behavior and models of such behavior. Other efforts have involved our continuing coordination with BBN, and some initial network experimentation and measurements using artificial traffic. Each of these efforts is discussed in more detail in the following paragraphs.

3.1. The Measurement and Control Program

The network measurement and control package was completed during the reporting period and is now operational. This program consists of a set of routines which (1) accept measurement data from the network, (2) control its processing and print out, (3) handle the initialization, control and termination of tests, and (4) provide a substantial message generation facility to generate artificial traffic. The data handling routine is a FORTRAN program which formats and prints out the measurement statistics generated by the IMP's. The program uses a number of assembly language subroutines which interface with the system monitor and control the interrupts, the clock and the IMP interface. In addition, the program can generate artificial traffic over a number of links and with a variety of message lengths and frequencies. The individual most active in this area at UCLA has been Gerald Cole, a graduate student on the contract.

The measurement program is controlled from the operator's console in a manner similar to that used to set up experiments at the IMP console. For example, the experimenter would type SNAP (NL) followed by 5 (NL) if he wished to have only every fifth snap-shot data message printed out. He has similar control capabilities for the accumulated statistics and trace data messages, as well as having control over the destination of the data (the line printer or magnetic tape), the source of the data (from the IMP or magnetic tape), and other optional features such as a decimal "dump" printout of the received data.

The message generator portion of the program allows the user to specify which of the 63 possible generators that he wishes to utilize, and to define the destination, the initial message length, the final message length, the increments for successive runs, the duration of the run in milliseconds, and the number of messages sent on each link. If a non-zero length increment has been specified, the experiment will terminate after the end message length has been run; but for a zero increment value, it will create continuous traffic until stopped from the operator's console. Statistics on the artificial traffic can also be printed out on the console at the operator's request.

Since the measurement program is of value to a number of experimental and research efforts, it has been documented in considerable detail, including a well commented program listing and a user guide

which was published as part of the Miscellaneous Network Note series.* The latter document lists and describes each of the thirty different commands including a precise format description as well as a description of what functions each command performs.

3.2. Analytical Efforts Related to Measurements

3.2.1. Data Analysis Efforts

Most of the data analysis effort has been devoted to the study of how one interprets data which has been obtained in logarithmic histogram formats, i.e. such as are used to measure packet and short message lengths. The approach has been to consider the logarithmic data as being the result of a transformation of variable from the actual variable of interest, x , to a new variable, ξ . Therefore, the desired information (the moments and/or density function) have been transformed into functions of the variable ξ , and we must perform the inverse transformation to obtain the desired data. Results were obtained relating the moments of the density $f(x)$ to the moments of the transformed function $\phi(\xi)$ but this relationship had very poor convergence properties, i.e. many higher moments in ξ were required to compute the moments in x . A second relationship was subsequently found using central moments which had better convergence properties. The analysis also included considerations of the grouped data aspects and resulted in the development of new moment correction formulas which compensate (on the average) for the fact that the

* Misc. Network Note #9, "Network Measurement Program", by D. Karas, 30 June 1970

data points are not uniformly distributed over a histogram interval.

A relationship was found which relates the moments of x directly to the log-histogram values, which when combined with the correction formulas, should provide good estimates of the moments from the relatively coarse histogram data.

3.2.2. Relations Between Measurements, Models, and the Actual System

An attempt was made to formalize the relationship between the measurements which one takes on a system, the models of the system, and the system itself. The system was considered to be represented by a set of structural relationships and a set of behavioral state variables. Measurements were then classified as being either subsets of the state variables or as a set of functions of present and past state variables, e.g. averages or histograms. The structural aspects are generally implicit in the measurements, e.g. the connectivity of IMP's as expressed by the modem output destinations.

Models of the system can vary considerably in the degree of detail which they include, and are not necessarily subsets or functions of a subset of the state variables. This difficulty leads us to at least temporarily abandon this approach as being a viable formalism, and subsequently it has been used merely as a classification vehicle for representing the various measurement techniques.

3.2.3. Control Aspects of Measurements

One of the functions of the measurement activity is to provide

insight into potential network improvements, fine-tuning, and perhaps eventually self-adaptive control. This necessary insight is developed by a combination of measuring and modelling efforts; and as an example, we are investigating the effects of varying the priority threshold. This threshold is presently set at a length of five IMP words, i.e. messages are considered to be of a higher priority if they do not exceed this length. Cox and Smith have considered this problem based on an exponential service time and found the optimal threshold to minimize the average queueing delay.* This work has been extended during the recent weeks to consider hyperexponential service because the user-to-computer and the computer-to-user message lengths have considerably different means, although we have assumed that each can be modelled separately by an exponential service time. More detailed models which consider the truncation effects of packet size segments are being developed.

3.3. Coordination With BBN

Coordination with BBN on the measurement effort has continued, but at a lesser degree since they have completed their portion of the measurement package. However, there have been some minor changes in the measurements such as deleting the ASCII conversion queue lengths from the snapshot formats and improving the resolution of the round-trip and arrival time data. We requested that they change these readings to utilize the 100 μ sec. clock instead of the 25.6 msec. clock, since the latter was comparable to the times that were being measured.

* Cox, D. R. and W. L. Smith, "Queues", Methuen's Monographs, 1961.

Various UCLA personnel cooperated with BBN in their network tests. Much of the artificial traffic capability discussed in section 3.1 was originally developed as part of that effort.

3.4. Experimentation

Some experiments have been run to exercise the network and to check-out the usage of the measurement routines. Figure 1 is a portion of an accumulated (12-Sec.) statistics printout from one of these tests, and shows the traffic that was established on each inter-IMP communication line. This particular test utilized several of the measurement messages as artificial traffic to supplement the IMP's pseudomessage generator. A snap-shot of the modem queues for a similar situation, but with a different subset of the IMP's is shown in Figure 2. A trace message from a third test is shown in Figure 3, and represents the trace data from a short (priority) message from UCLA to SRI. The message was sent via UCSB because the channel 1 modem at SRI was looped.

Other tests had been run to debug our measurement routines and to gain familiarity with interpreting the measurement results, but some of the tests provided some interesting results in themselves. For example, the propagation delay between UCSB and SRI was calculated from trace data and was felt to be too high. We later found that the line between UCSB and SRI was routed via Los Angeles, which explained the slight extra delay.

4. CHANNEL ACTIVITY

	# HELLO SENT	# IHY RECVD	# ACK SENT	# ACK RECVD	# PKTS RECVD	# INPUT ERRORS
CHANNEL 1	10	10	1	56	45	0
CHANNEL 2	1	1	37	0	114	0
CHANNEL 3	20	0	0	0	0	0
TOTALS	31	11	38	56	159	0

- VERY LITTLE TRAFFIC ON CH 1
- NORMAL TRAFFIC ON CH 2
- NO RESPONSE ON CH 3

	# RE- TRANS.	# RFNM SENT	# WORDS SENT	FREE LIST EMPTY	# BUFFERS REMOVED
CHANNEL 1	0	0	154	0	0
CHANNEL 2	0	0	1475	0	0
CHANNEL 3	0	0	0	0	0
TOTALS	0	0	1629	0	0

14

5. PACKET SIZE STATISTICS (LEAVING THE IMP)

LENGTH (WORDS)

0 - 1	0
2 - 3	0
4 - 7	0
8 - 15	0
16 - 31	1
32 - 63	2
TOTAL # PKTS	3

CHANNEL 2

0	*
1	*
15	*
5	*
16	*
16	*
53	*

THE 12-SEC DATA
MESSAGE -
2 63 WORD PKTS +
1 28 WORD PKT

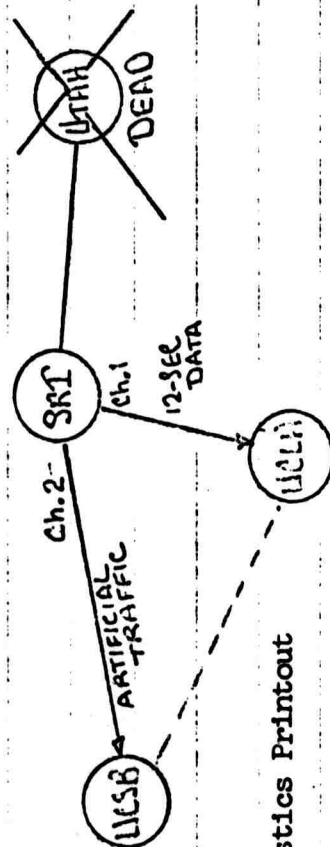


Figure 1. A Portion of an Accumulated Statistics Printout

SNAPSHOT STATISTICS

SOURCE = UCLA TIME = 38012 DATE = 1-27-70 PRINT FREQUENCY = 20

1. HOST QUEUE LENGTHS

	HOST 0	HOST 1	HOST 0	HOST 1	HOST 2	HOST 3
NORMAL MESSAGE QUEUE	0	0	0	0	0	0
PRIORITY MESSAGE QUEUE	0	0	0	0	0	0

2. CHANNEL QUEUE LENGTHS

OUTPUT Q RFNM Q PRIORITY ACK Q SENT Q

CHANNEL 1	0	0	0	0	0
CHANNEL 2	4	0	0	1	0
CHANNEL 3	0	0	0	0	0

3. ROUTING TABLE INFORMATION

DESTINATION HOP # DELAY BEST CHAN. HOST ALIVE

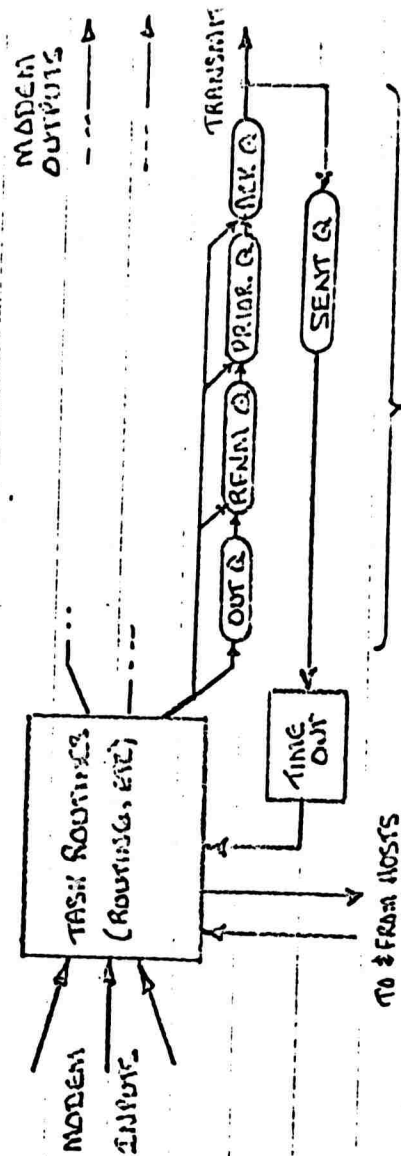
UCLA	0	0	1	YES
SRI	31	2047	0	YES
UCSB	1	28	2	YES
UTAH	31	2047	0	YES

4. BUFFER STORAGE

FREE STORAGE LIST LENGTH = 23
NO. OF STORE & FWD BUFFERS = 5
NO. OF MES. REASSY BUFFERS = 8
TOTAL NO. OF BUFFERS = 36

5. LINE ALIVE COUNTS

CHNL 1	CHNL 2	CHNL 3
0	43	0



TYPICAL SET OF CHANNEL QUEUES

Figure 2. A Snap Shot Printout

TRACE DATA SOURCE = UCSB DATE = 2- 3-70 OVERFLOW : NONE PRINT FREQUENCY = 1

MSG. #	TYPE	SOURCE	DESTINATION	FUNCTION	T(IN)	T(QUEUE)	T(XMIT)	T(*)
23	REGULAR	GHOST 3,UCLA	GHOST 3,SRI	STOR&FW	427.1	427.3	427.7	453.6

MSG. #	LINK #	PKT #	LAST PKT	PRIORITY	CHANNEL	STATUS
23	6	0	YES	NB	2	ACK

T(*) IS TIME TRANSMISSION TO HOST COMPLETED FOR IMP TO HOST MESSAGES
- OTHERWISE TIME ACKNOWLEDGMENT RECEIVED

Figure 3. A Trace Message Printout

4. NETWORK AND SYSTEMS SOFTWARE

This section covers work done in the Spade group during the first half of calendar year 1970. The larger portion of our effort has gone into fixing and extending the SEX^{*} time-sharing system, the remainder of our effort has been on the network. In terms of progress, however, the network effort has been much more visible.

4.1. Network Progress

Following a mandate to change the protocol so that it would permit access to a Host without logging into it, the Network Work Group developed the present and more powerful protocol. UCLA has been a major participant in the Network Working Group, with Steve Crocker becoming chairman in May. A paper authored by S. Carr, S. Crocker and V. Cerf [11] was presented at the SJCC in May. Network meetings were held on March 17 at UCLA, May 7 in Atlantic City, May 8 at Lincoln Laboratories, June 29 at Harvard and July 1 at UCLA, and Network Working Group/Request for Comments numbers 28 through 59 were written and distributed. About a third of these NWG/RFC's were written at UCLA; eight other sites contributed the remainder.

The outcome of this effort is the agreement on a Host-Host protocol, substantial progress in the development of the next layer of protocol, and explanation of some of the very hard problems in using complex

^{*}Sigma EXecutive.

interactive subsystems over the network. NWG/RFC's 33, 36-40, 44, 46-50, 54, 55, 57-59 relate to Host-Host protocol, NWG/RFC's 42 and 56 cover the next layer, and NWG/RFC's 31 and 51 pertain to higher level languages to deal with complex subsystems.

In addition to the effort on the general Host-Host protocol, we engaged in an experiment with SRI and another with Utah. These experiments showed that all of the operating systems contained deficiencies which prohibited convenient network operation. These experiments did not use the Host-Host protocol. These deficiencies were that multiple layers of control character scanning on the PDP-10 prohibited getting necessary control characters, e.g. a carriage return down to the subsystem, and on the 940 it was not possible to log-in. On our system, it was very painful to operate in character-oriented mode. These systems are all undergoing modification to correct these problems; moreover, the Host-Host protocol provides a general solution to some of the problems.

Finally, during March we provided programming support to the BB&N staff for their network experiments conducted at UCLA. We are also supporting Jerry Cole's measurement work.

4.2. System Development

The GORDO time-sharing system acquired from LRL at Livermore required major modifications and extensions at the beginning of this year; the system has been modified to run on our hardware, and extensions to the system are underway. The important differences between our

hardware and Livermore's are disk geometry, consoles and number of register blocks. The extensions to the system which have been completed are the inclusion of a background batch processor, the modification of the terminal handler to treat a variety of terminals and to provide for both character-oriented and line-oriented communication, and the construction of interactive utilities for moving, listing and deleting files and starting, stopping and examining processes. Current facilities also include FORTRAN, an assembler and loader, Meta5, and a very major effort has also been expended on documentation. The status of the system is detailed in notebook form and runs to about 350 pages. Design changes and preliminary specifications of new facilities are documented as Spade Design Notes; Spade Design Notes 5 through 25 appeared in the first half of the year. More formal user documentation manual on Botch, the background batch processor, has been produced.

The changes in the time-sharing system have been extensive enough to merit changing the name. Taking a cue from BB&N which is producing ~~TENEX~~, for TEN EXecutive, our system for the Sigma Seven is SEX. The SEX system is currently running 2-6 hours per day and is servicing the Spade group. Full time operation is projected to begin after September 1 and before December 31. Implementation of the protocol should be complete by September 5.

5. CONCLUSIONS AND SELF-EVALUATION

We feel that the effort expended in the modelling and analytical area has paid off handsomely and will continue to develop and grow in the directions indicated in this progress report. Our measurement effort has been moving slowly as the network itself has unfolded during these last few months. That effort is now taking a step function increase in activity and extensive tests using artificial traffic are now being implemented. When real traffic begins to throb in the network, then that measurement activity will shift over and measure the actual traffic conditions. We have put considerable effort into the network and systems software, and progress in that area has been substantial. The network protocol is now moving along very nicely and should peak out sometime within the next six months. The same may be said about our time-sharing system, SEX. The effort has been very large, and we are just now beginning to reap the benefits of that investment. We hope to be able to shift the emphasis from the hard core programming required of these last two tasks into the modelling and analysis area.

As a result of our efforts, UCLA now has an excellent reputation in the area of modelling and analysis among computer scientists and is an acknowledged and important participant in the ARPA Network community.

REFERENCES

BLANK PAGE

REFERENCES

1. Kleinrock, L. "Time-Sharing Systems: Analytical Methods," Proc. of the Symposium on Critical Factors in Data Management/1968, UCLA, March 20-22, 1968, Prentice-Hall, pp. 3-32, 1969.
2. Martin, D., and G. Estrin. "Path Length Computations on Graph Models of Computations," Transactions of the IEEE, Vol. C-18, pp. 530-536, June 1969.
3. Kleinrock, L. "Models for Computer Networks," Proc. of the IEEE International Conference on Communications, Boulder, Colo., pp. 21-16 to 21-29, June 9-11, 1969.
4. Kleinrock, L. "On Swap Time in Time-Shared Systems," Proc. of the IEEE Computer Group Conference, Minneapolis, Minn., pp. 37-41, June 17-19, 1969.
5. Coffman, E. G., Jr., and R. R. Muntz. "Model of Pure Time-Sharing Disciplines for Resource Allocation," Proc. of the 24th National Conference of ACM, August 1969.
6. Chu, W. W. "A Study of Asynchronous Time Division Multiplexing for Time-Sharing Computer Systems," 1970 Proc. of the Fall Joint Computer Conference, Las Vegas, Nev., pp. 669-678, November 1969.
7. Kleinrock, L. "Comparison of Solution Methods for Computer Network Models," Proc. of the Computer and Communications Conference, Rome, N.Y., Oct. 2, 1969.
8. Muntz, R. R., and R. Uzgalis. "Dynamic Storage Allocation for Binary Search Trees in a Two-Level Memory," Proc. of the Fourth Annual Princeton Conference on Information Sciences and Systems, Princeton, N.J. March 26-27, 1970.
9. Kleinrock, L. "A Continuum of Time-Sharing Scheduling Algorithms," Proc. of the 1970 Spring Joint Computer Conference, Atlantic City, N.J., pp. 453-458, May 1970.
10. Kleinrock, L. "Analytic and Simulation Methods in Computer Network Design," Proc. of the 1970 Spring Joint Computer Conference, Atlantic City, N.J., pp. 569-579, May 1970.
11. Carr, C. S., S. D. Crocker, and V. G. Cerf, "HOST-HOST Communication Protocol in the ARPA Network," Proc. of the 1970 Spring Joint Computer Conference, Atlantic City, N.J., pp. 589-597, May 1970.
12. Chu, W. W. "Selection of Optimal Transmission Rate for Statistical Multiplexors," Proc. of the 1970 IEEE International Conference on Communications, San Francisco, Calif., pp. 28-22 to 28-25, June 8-10, 1970.

13. Kleinrock, L. "Swap Time Considerations in Time-Shared Systems," IEEE Transactions on Computers, pp. 534-540, June 1970.
14. Kleinrock, L., and R. R. Muntz. "Multilevel Processor-Sharing Queueing Models for Time-Shared Systems," Proc. of the Sixth International Teletraffic Congress, Munich, Germany, pp. 341/1-341/8, August 1970.
15. Cantor, D. "Non-Blocking Switching Networks," to appear in J. Networks, Vol. I, Issue 1, 1970.
16. Kleinrock, L., and R. R. Muntz, and E. Rodemich. "The Processor-Sharing Queueing Model for Time-Shared Systems with Bulk Arrivals," to be published.
17. Kleinrock, L., and R. R. Muntz. "Processor-Sharing Queueing Models of Mixed Scheduling Disciplines," to be published.
18. Chu, W. W. "Buffer Behavior for Batch Poisson Arrivals and Single Constant Output," to be published in IEEE Trans. on Communication Technology, October 1970.

Presentations

1. Kleinrock, L. "Analytic Methods and Results for Time-Shared Systems: A Survey," presented at Princeton University, Princeton, N.J., October 17, 1969.
2. Kleinrock, L. "Mathematical Analysis of Time-Shared Scheduling Algorithms," presented at University of Utah, Salt Lake City, December 5, 1969.
3. Kleinrock, L. "Queueing Theory, Computer Networks, and Time-Shared Systems," presented at Network Analysis Corp., Glen Cove, N.Y., January 21, 1970.
4. Kleinrock, L. "Analytic Models for Computer Systems," presented to the SIAM meeting at The RAND Corp., Santa Monica, Calif., April 11, 1970.
5. Kleinrock, L. "Measurement and Modelling of the ARPA Computer Network," presented at the Operations Research Society of America Conference, Washington, D.C., April 20, 1970.
6. Kleinrock, L. "Time-Shared Computer Systems Analysis," presented at Brown University, Providence, R.I., May 8, 1970.
7. Kleinrock, L. "Analytic Models for Time-Shared Computer Systems," presented at IBM, San Jose, Calif., June 8, 1970.
8. Kleinrock, L. "A Selected Menu of Analytic Results for Time-Shared Computer Systems," to be presented at IBM, Germany, September 18, 1970.

A P P E N D I C E S

APPENDIX A

SWAP-TIME CONSIDERATIONS IN TIME-SHARED SYSTEMS

by

Leonard Kleinrock

normalized queuing delay due to buffering is equal to 1.25 character-service times. Since each service time equals $1/\mu = 1/240 = 4.16$ ms, the waiting time of each character is 5.06 ms. Now suppose the number of terminals increases from 48 to 96, so that the traffic intensity is less than unity, two transmission lines are needed, and the traffic intensity is still equal to 0.6. From Fig. 2, the buffer length corresponding to the desired overflow probability for two transmission lines is about 14 characters. The waiting time is about 0.8 character-service times which is equal to 3.33 ms. Although the difference between 5.06 and 3.33 ms may not be detected by a user at a terminal, a common buffer of the same size operating with two output lines can handle twice the number of input lines as with one output line. Thus, the common buffer approach permits handling a wide range of traffic without substantial variation in buffer size.

REFERENCES

- [1] W. W. Chu, "A study of asynchronous time division multiplexing for time-sharing computer communications," presented at the 2nd Hawaii Internatl. Conf. System Sciences (Honolulu, Hawaii), January 22-24, 1969.
- [2] B. A. Powell and B. Avi-Itzhak, "Queuing systems with enforced idle time," *Operations Res.*, vol. 15, no. 16, pp. 1145-1156, November 1967.
- [3] T. G. Birdsall, M. P. Ristenbatt, and S. B. Weinstein, "Analysis of asynchronous time multiplexing of speech sources," *IRE Trans. Communications Systems*, vol. CS-10, pp. 390-397, December 1962.
- [4] N. M. Dor, "Guide to the length of buffer storage required for random (Poisson) input and constant output rates," *IEEE Trans. Electronic Computers* (Short Notes), vol. EC-16, pp. 683-684, October 1967.
- [5] J. D. C. Little, "A proof of the queuing formula $L = \lambda w$," *Operations Res.*, vol. 9, pp. 383-387, 1961.
- [6] R. W. Hamming, *Numerical Methods for Scientists and Engineers*. New York: McGraw-Hill, 1962, pp. 363-364.
- [7] P. M. Morse, *Queues, Inventories and Maintenance*. New York: Wiley, 1958, pp. 15-18.
- [8] W. W. Chu, "Optimal file allocation in a multiple computer system," *IEEE Trans. Computers*, vol. C-18, pp. 885-889, October 1969.

Swap-Time Considerations in Time-Shared Systems

LEONARD KLEINROCK, MEMBER, IEEE

Abstract—Solved for is the expected swap time expended for those customers in the system of queues in general models of time-shared systems. This quantity is expressed in terms of the expected queuing time conditioned on required service time and is applied to a number of examples of interest.

Index Terms—Modeling and analysis, processor-sharing, queuing analysis, scheduling, swap time, time-shared.

INTRODUCTION¹

NUMEROUS authors have addressed themselves to the problem of solving for the average response time T in time-shared computer systems [1]-[11] and an excellent summary of such investigations is available [9]. Many of these studies condition T on the required service time (i.e., the required processing time) t which we denote by $T(t)$. Some go further and introduce an external priority system, solving for $T_p(t)$ which is the average response time for a customer from priority group p who requires t seconds of processing time [4].

Recently a new quantity, the distribution of attained service time $N_p(\tau)$ was calculated [6] for any priority

feedback queueing system which satisfies Little's result (which states that the average number to be found in the system is equal to the average arrival rate of customers times the average time spent in that system). $N_p(\tau)$ is defined as the expectation of the number of customers in the system of queues from priority group p who have so far received exactly τ seconds of useful processing. A customer is said to be in the system of queues whenever he is waiting for his next quantum of service time during his request for t seconds of total service. We assume that on his n th visit to service, a customer from priority group p will receive the attention of the processing unit for $g_{pn}Q$ seconds.

RESULTS

In this paper, we are interested in swap-time considerations. (Swap time is the time spent in removing the old customer from and bringing the new customer into service, as well as any other cost in time directly related to this operation.) Our main result is that a measure of this quantity is simply expressed in terms of $N_p(\tau)$ as follows. We direct our attention to the customers in the system of queues and we inquire as to the expected time S , which has been expended in swapping for this set of customers. The answer comes directly from $N_p(\tau)$. First we note that all customers from group p who have visited the service facility exactly n times

Manuscript received August 14, 1969; revised December 16, 1969. This work was supported by the Advanced Research Projects Agency of the Department of Defense under Contract DAI1C15-69-C-0285. Reproduction in whole or in part is permitted for any purpose of the United States Government.

The author is with the School of Engineering and Applied Science, University of California, Los Angeles, Calif. 90024.

¹ This paper evolved from and is an extension of the paper in [3].

KLEINROCK: SWAP-TIME CONSIDERATIONS

must so far have received useful processing in an amount equal to

$$\tau_n = \sum_{i=1}^n (g_{pi}Q - \theta_{pi}) \text{ seconds} \quad (1)$$

where θ_{pn} = (wasted) swap time used for a customer from group p on his n th visit to service.² Thus only τ_n will appear as a meaningful argument for $N_p(\tau)$. Clearly, then, the expected swap time expended for all customers from group p who are in the system of queues must be

$$S_p = \sum_{n=1}^{\infty} \gamma_{pn} N_p(\tau_n) \quad (2)$$

where

$$\gamma_{pn} = \sum_{i=1}^n \theta_{pi} \quad (3)$$

But from Theorem 1 of [6] we have that

$$N_p(\tau_n) = \lambda_p [1 - B_p(\tau_n)] [W_p(\tau_{n+1}) - W_p(\tau_n)] \quad (4)$$

where

λ_p = average arrival rate of customers from group p ,

$B_p(t)$ = P (required processing time for customers from p th priority group $\leq t$), and

$W_p(t)$ = expected wait in queues for customers from group p who require a total of t seconds of processing.

Finally, to solve for S (the expected swap time expended on all customers in the system of queues) we have the following.

Theorem 1: For time-shared systems ($Q > 0$)

$$S = \sum_{p=1}^P S_p = \sum_{p=1}^P \sum_{n=1}^{\infty} \gamma_{pn} \lambda_p [1 - B_p(\tau_n)] [W_p(\tau_{n+1}) - W_p(\tau_n)] \quad (5)$$

where

P = total number of priority groups.

Observe that S is expressed in terms of known quantities (γ_{pn} , λ_p , $B_p(\tau_n)$) and a function $W_p(\tau_n)$ which is the average conditional waiting time; this last measure is the usual one solved for in the analysis of time-shared systems.

Corollary 1: For $\theta_{pn} = \theta_p$ independent of n ,

$$S = \sum_{p=1}^P \lambda_p \theta_p \left[\sum_{n=1}^{\infty} n b_p(\tau_{n+1}) W_p(\tau_{n+1}) - \sum_{n=1}^{\infty} [1 - B_p(\tau_n)] W_p(\tau_n) \right] \quad (6)$$

² We assume the obvious condition that $g_{pn}Q \geq \theta_{pn}$ for all n .

where

$$\begin{aligned} b_p(\tau_{n+1}) &= B_p(\tau_{n+1}) - B_p(\tau_n) \\ &= P[\text{customer from group } p \text{ has required service time } t \text{ such that } \tau_n < t \leq \tau_{n+1}] \\ &= P[\text{customer from group } p \text{ requires exactly } n+1 \text{ visits to the service facility}]. \end{aligned}$$

Proof: We have $\gamma_{pn} = n\theta_p$. Substituting this in (5) and using $1 - B_p(\tau_n) = [1 - B_p(\tau_{n+1})] + [B_p(\tau_{n+1}) - B_p(\tau_n)]$ we get (6).

Corollary 2: For $\theta_{pn} = \theta_p$ and $P = 1$ (i.e., no priorities and so we drop the subscript p) we have

$$S = \lambda \theta \sum_{n=1}^{\infty} n b(\tau_{n+1}) W(\tau_{n+1}) - \lambda \theta \sum_{n=1}^{\infty} [1 - B(\tau_n)] W(\tau_n). \quad (7)$$

Proof here is immediate from Corollary 1.

We now consider Theorem 1 plus its corollaries for the *processor-shared systems* [4]. These systems are time-shared systems in which the quantum size Q is allowed to shrink to zero. In this limit for $Q \rightarrow 0$, we must obviously contain the swap time θ_{pn} in a meaningful way such that $g_{pn}Q \geq \theta_{pn}$. This we do in the (theoretically) natural way, namely, defining

$$\phi_{pn} = \frac{\theta_{pn}}{g_{pn}Q} \quad (8)$$

where we require $0 \leq \phi_{pn} \leq 1$. Thus ϕ_{pn} is that *fraction* of the n th quantum given to a customer from group p which is wasted due to swap time. In the limit as $Q \rightarrow 0$, we must then define

$$\phi_p(\tau) = \lim_{Q \rightarrow 0} \phi_{pn} \quad (9)$$

where for a given p and τ , we consider an $n = n(Q)$ increasing as Q decreases such that

$$\tau = \lim_{Q \rightarrow 0} \sum_{i=1}^{n(Q)} (g_{pi}Q - \theta_{pi}) = \lim_{Q \rightarrow 0} \sum_{i=1}^{n(Q)} g_{pi}Q(1 - \phi_{pi}). \quad (10)$$

As discussed in [4], this $Q \rightarrow 0$ limit has useful characteristics in our analysis of time-shared systems. Developing the analogous equations here, we have

$$S_p = \int_0^{\infty} \gamma_p(\tau) \cdot W_p(\tau) d\tau \quad (11)$$

where

$$\gamma_p(\tau) = \int_0^{\tau} \phi_p(t) dt. \quad (12)$$

Note that $\gamma_p(\tau)$ is the time wasted in providing τ seconds of useful service to a customer from group p . But, from Theorem 2 of [6], we have that

$$N_p(\tau) = \lambda_p [1 - B_p(\tau)] \frac{dW_p(\tau)}{d\tau} \quad (13)$$

where λ_p , $B_p(t)$, and $W_p(t)$ are as defined above and $N_p(\tau)$ is now an expected density function.

We thus have S for this case as follows.

Theorem 2: For processor-shared systems ($Q \rightarrow 0$)

$$S = \sum_{p=1}^P S_p = \sum_{p=1}^P \int_0^\infty \gamma_p(\tau) \lambda_p [1 - B_p(\tau)] \cdot \left(\frac{dW_p(\tau)}{d\tau} \right) d\tau. \quad (14)$$

This last may be interpreted as a Stieltjes integral in the case that $W_p(\tau)$ is discontinuous.

Corollary 1: For $\phi_p(\tau) = \phi_p$ independent of τ , we have

$$S = \sum_{p=1}^P \lambda_p \phi_p \int_0^\infty \tau [1 - B_p(\tau)] \left(\frac{dW_p(\tau)}{d\tau} \right) d\tau. \quad (15)$$

Proof here is immediate since $\gamma_p(\tau) = \tau \phi_p$ by (12).

Corollary 2: For $\phi_p(\tau) = \phi_p = \phi$ (i.e., no dependence on τ and $P = 1$ giving no priorities thus allowing us to drop all subscripts) we have

$$S = \lambda \phi \int_0^\infty \tau [1 - B(\tau)] \left(\frac{dW(\tau)}{d\tau} \right) d\tau. \quad (16)$$

(Observe that (11) through (16) for the processor-sharing case are analogous to (2) through (7) for the time-sharing case.) It is important to note in this last (simplest) case that ratio of S to $\bar{\tau}$ (the average attained service) is given simply as follows: $\bar{\tau}$ is defined by

$$\bar{\tau} = \frac{\int_0^\infty \tau N(\tau) d\tau}{\int_0^\infty N(\tau) d\tau} \quad (17)$$

where the denominator is merely \bar{N} = expected number of customers in the system of queues. Since, from (11) and for $P = 1$, we have

$$S = \int_0^\infty \phi \tau N(\tau) d\tau,$$

we then obtain

$$\frac{S}{\bar{\tau}} = \phi \bar{N} \quad (18)$$

which clearly represents the ratio of average (wasted) swap time to average (useful) attained service time for the set of customers still in the system of queues. The simplicity and intuitive appeal of this last equation further supports the utility of using processor-shared models.

EXAMPLES

In order to apply the results obtained above, we must find in the published literature solutions for $W_p(t)$ (the expected wait conditioned on a service requirement of t seconds) for time-shared systems which account for swap time. Such results are not especially numerous. We do, however, find some useful examples.

Example 1: Let us apply Corollary 2 of Theorem 1 to

the discrete round robin infinite-input population system [5] where

$$b(\tau_{n+1}) = (1 - \sigma) \sigma^n \quad n = 0, 1, 2, \dots; \quad 0 \leq \sigma < 1 \quad (19)$$

$$P[1 \text{ arrival in interval of length } Q] = \lambda Q;$$

$$0 \leq \lambda Q < 1$$

$$P[0 \text{ arrival in interval of length } Q] = 1 - \lambda Q$$

$$g_{pn} = 1; \quad P = 1$$

where $P[x]$ is read "probability of x ." This system has an exact solution for $W(\tau_n)$, (for $\theta = 0$), and also a simple approximation to $W(\tau_n)$ given below (see [5]):

$$W(\tau_n) = W(nQ) \cong \frac{\rho n Q \sigma}{1 - \rho} \quad (20)$$

where

$$\rho = \lambda Q / (1 - \sigma). \quad (21)$$

When we consider $\theta > 0$, a number of considerations enter. The major question is now to keep the discrete model intact since, for example, if $\theta = Q/3$, then a customer requiring Q seconds of useful processing (one quantum for $\theta = 0$) now requires a noninteger number of quanta. No satisfactory way appears to resolve this, and so we will take two approaches. We begin with a continuous distribution of service time, namely the exponential, where $P[\text{service time} \leq t] = 1 - e^{-\mu t}$. We then recognize that all customers with $n(Q - \theta) < t \leq (n+1)(Q - \theta)$ will need exactly $(n+1)$ visits to the service facility (we make the simplifying, but serious assumption that unused portions of quanta are lost) and the probability that a new arrival satisfies such a condition is

$$b(\tau_{n+1}) = \int_{n(Q-\theta)}^{(n+1)(Q-\theta)} \mu e^{-\mu t} dt = (1 - e^{-\mu(Q-\theta)}) [e^{-\mu(Q-\theta)}] n.$$

Comparing this to (19), we make the correspondence

$$\sigma = e^{-\mu(Q-\theta)}. \quad (22)$$

We then apply (20) and (22) to Corollary 2 of Theorem 1 to give for this example,

$$S = \lambda \theta \sum_{n=1}^{\infty} n (1 - \sigma) \sigma^n \rho n Q \sigma / (1 - \rho) \\ - \lambda \theta \sum_{n=1}^{\infty} \sigma^{n+1} \rho n Q \sigma / (1 - \rho).$$

This gives

$$S = \frac{\lambda \theta \rho Q \sigma^2}{(1 - \rho)(1 - \sigma)^2} \quad (23)$$

where ρ and σ are given by (21) and (22), respectively. Note that for $S < \infty$, we require $\rho < 1$ which requires

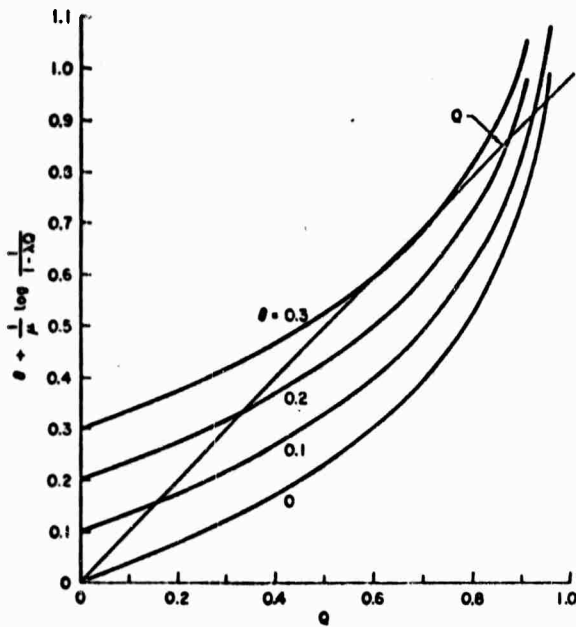


Fig. 1. Allowable values for Q shown as that region for which the curve $\theta + (1/\mu) \log 1/(1-\lambda Q)$ lies below Q . ($\lambda=1, \mu=3$) for Example 1.

$$Q > \theta + \frac{1}{\mu} \log \frac{1}{1-\lambda Q}. \quad (24)$$

Equation (24) places lower and upper bounds on Q . The lower bound is due to the restriction that the maximum effective service rate must exceed the average arrival rate. The upper bound is due to the wasted excess quantum and also due to the constraint that the arrival probability $\lambda Q < 1$.

In Fig. 1 we show the allowed range of Q as determined from (24) for $\lambda=1, \mu=3$, and θ as parameter. Fig. 2 gives S as a function of Q (in its allowed range) with θ as parameter again and for the same values of λ and μ . Here we also plot the locus of optimum Q over the family of θ curves to give minimum S .

The second approach to handling the $\theta > 0$ case in the first example is to allow the exponential distribution of required service time to remain, but not force it into the discrete form of (19). We then get an equivalent system with $\theta=0$ if we segment this distribution into pieces of width $Q-\theta$, each separated by a gap of size θ as shown in Fig. 3. This results in a mean service time $E(t)$ and a second moment $E(t^2)$ given by

$$E(t) = \frac{1}{\mu} + \frac{\theta}{1 - e^{-\mu(Q-\theta)}}$$

$$E(t^2) = \frac{2}{\mu^2} + \frac{\theta\left(\theta + \frac{2}{\mu}\right)}{1 - e^{-\mu(Q-\theta)}} + 2Q\theta \frac{e^{-\mu(Q-\theta)}}{[1 - e^{-\mu(Q-\theta)}]^2}$$

One may now use this service-time distribution in the continuous round robin model studied in [2], as well as in other models of time-shared systems, with additional care to replace Q by $Q-\theta$ in the appropriate places.

We do not carry out this exercise here.

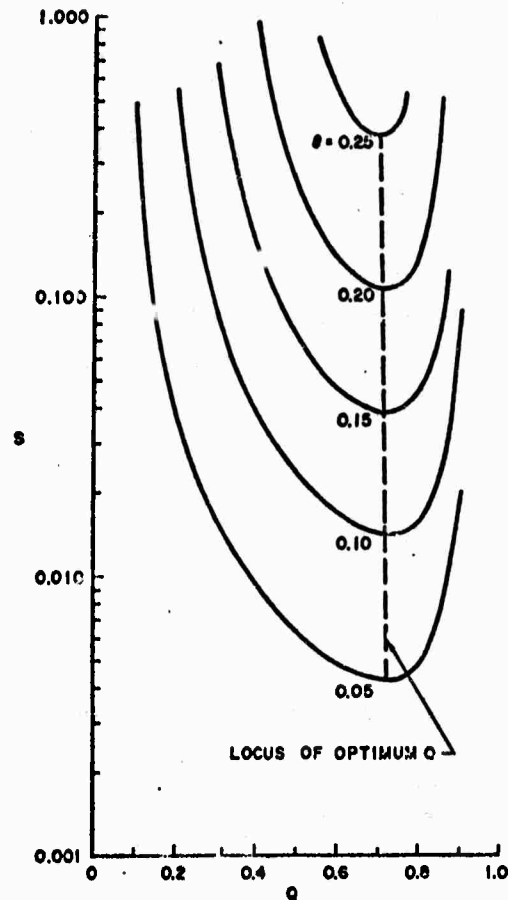


Fig. 2. Average swap time S as a function of quantum size Q with swap time (per quantum) θ as a parameter for Example 1 ($\lambda=1, \mu=3$).

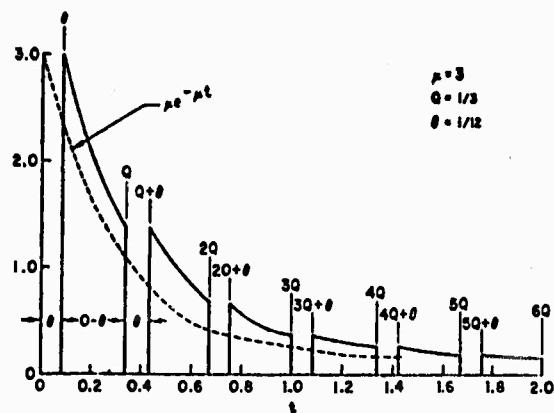


Fig. 3. Segmented exponential service-time distribution to account for swap time θ .

The following examples apply only to the processor-sharing case.

Example 2: Consider the continuous (processor-sharing) round robin infinite-population system [4]. Again we assume $g_{pn}=1, P=1$. Here we have Poisson arrivals at rate λ and exponential service of average duration $1/\mu$. For this system we know from [4] that for $\phi=0$,

$$W(r) = \rho r / (1 - \rho) \quad (25)$$

where

$$\rho = \lambda/\mu.$$

The original exponential distribution of service time with parameter μ for this system must now be replaced with another exponential distribution with parameter $\mu(1-\phi)$ where ϕ = fraction of service time wasted (as above).

We now apply Corollary 2 of Theorem 2 to give

$$S = \lambda\phi \int_0^{\infty} \tau e^{-\mu(1-\phi)\tau} [\rho/(1-\rho)] d\tau.$$

This results in

$$S = \frac{\rho^2\phi}{\mu(1-\rho)(1-\phi)} \quad (26)$$

where

$$\rho = \lambda/\mu(1-\phi). \quad (27)$$

The average swap time S is plotted in Fig. 4 versus ϕ with λ/μ as parameter for this example.

It is interesting to note the application of (18) here. From [4] we have

$$\bar{N} = \rho^2/(1-\rho)$$

and from [6] we have (with the new value $\mu(1-\phi)$)

$$\bar{\tau} = 1/\mu(1-\phi).$$

Thus

$$\begin{aligned} S &= \phi \bar{N} \bar{\tau} \\ &= \frac{\rho^2\phi}{\mu(1-\rho)(1-\phi)} \end{aligned}$$

which is (26) again.

Further, we can calculate this by considering the average swap time per customer, S' . S' is the product of average service time ($=1/\mu(1-\phi)$) and the fraction of wasted (swap) time ($=\phi$). Thus, $S' = \phi/\mu(1-\phi)$. This multiplied by \bar{N} must also give S , as is obvious.

We now show that the result of Example 1 can be taken to the limit of $Q=0$ with fixed $\phi=\theta/Q$ to give the result of Example 2. From Example 1, we get as $Q \rightarrow 0$

$$\begin{aligned} \rho &= \lambda Q/(1-\sigma) \\ &= \lambda Q/(1-e^{-\mu(Q-\theta)}) \\ &= \lambda Q/(1-1+\mu(Q-\theta)-\dots) \\ &= \lambda/\mu(1-\phi). \end{aligned}$$

Thus (21) limits to (27). Also, as $Q \rightarrow 0$

$$\begin{aligned} S &= \frac{\lambda\theta\rho Q\sigma^2}{(1-\rho)(1-\sigma)^2} \\ &= \frac{\lambda\left(\frac{\theta}{Q}\right)\left(\frac{\lambda}{\mu(1-\phi)}\right)Q^2e^{-2\mu(Q-\theta)}}{\left(1-\frac{\lambda}{\mu(1-\phi)}\right)(1-e^{-\mu(Q-\theta)})^2} \end{aligned}$$

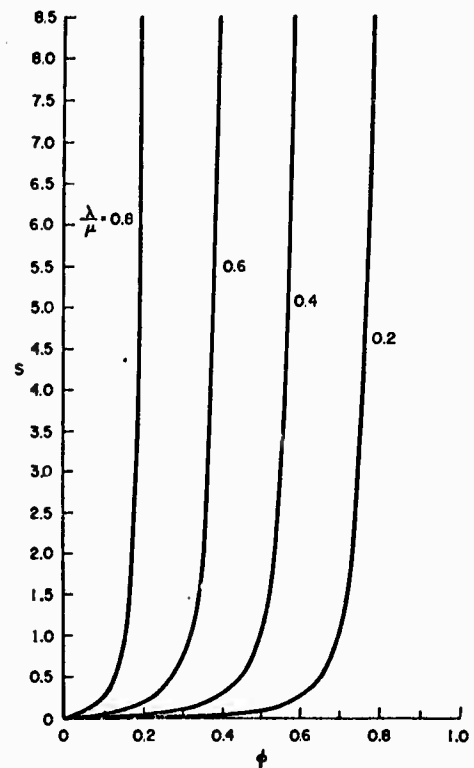


Fig. 4. Average swap time S as a function of percentage swap time ϕ with λ/μ as parameter for Example 2 ($\mu=3$).

$$\begin{aligned} &= \frac{\lambda^2\phi Q^2}{[\mu(1-\phi)-\lambda][1-1+\mu(Q-\theta)-\dots]^2} \\ &= \frac{\lambda^2\phi}{[\mu(1-\phi)-\lambda][\mu(1-\phi)]^2} \\ &= \frac{\rho^2\phi}{\mu(1-\phi)(1-\rho)}. \end{aligned}$$

Thus (23) limits to (26).

Furthermore, if we wish to find the average swap time S_T for all customers still in system (queues plus service), we merely replace $\bar{N} = \rho^2/(1-\rho)$ with $\bar{N}_{TOTAL} = \rho/(1-\rho)$ which then gives $S_T = \rho\phi/[\mu(1-\rho)(1-\phi)]$.

Example 3: Here we consider the priority processor-shared case studied in [4]. We have $g_{pn} = g_p$, $B_p(t) = 1 - e^{-\mu_p t}$, and Poisson arrivals at rate λ_p for group p , $p=1, 2, \dots, P$. Allowing swap time of form $\theta_{pn} = \theta_p$, we recognize that we must modify the service time distribution to take the form $B_p(t) = 1 - e^{-\mu_p(1-\phi_p)t}$. From [4] we get

$$W_p(\tau) = \frac{\tau}{g_p(1-\rho)} \sum_{i=1}^P g_i \rho_i \quad (28)$$

where

$$\rho_p = \lambda_p/\mu_p(1-\phi_p). \quad (29)$$

Applying Corollary 1 of Theorem 2 we get

$$S = \sum_{p=1}^P \frac{\lambda_p \phi_p}{g_p(1-\rho)} \int_0^{\infty} \tau e^{-\mu_p(1-\phi_p)\tau} \left(\sum_{i=1}^P g_i \rho_i \right) d\tau$$

$$S = \sum_{p=1}^P \frac{\rho_p \phi_p \sum_{i=1}^P g_i \rho_i}{g_p(1-\rho)\mu_p(1-\phi_p)} \quad (30)$$

where

$$\rho = \sum_{p=1}^P \rho_p.$$

In order to plot S , we must choose values for $\{g_p\}$, $\{\mu_p\}$, $\{\lambda_p\}$, and $\{\phi_p\}$. For the case $\lambda_p = \lambda/P$, $\mu_p = \mu$, $\phi_p = \alpha/g_p$ (where $0 \leq \alpha \leq 1$) one obtains from (30),

$$S = \left(\frac{\alpha \lambda^2}{\mu^2 P^2} \right) \frac{\sum_{p=1}^P \frac{1}{(g_p - \alpha)^2} \sum_{i=1}^P \frac{g_i^2}{g_i - \alpha}}{1 - \frac{\lambda}{\mu P} \sum_{j=1}^P \frac{g_j}{g_j - \alpha}}. \quad (31)$$

This last is plotted in Fig. 5 versus α with $\mu=3$, $\lambda=1$, $P=5$, and for the following eight cases: $g_p=1$, $g_p=(1.01)^p$, $g_p=(1.1)^p$, $g_p=\log_2(p+1)$, $g_p=p$, $g_p=p^{3/2}$, $g_p=p^2$, $g_p=p^{5/2}$. Note the interesting effect where S decreases as we progress through the first four cases and then increases as we progress through the last four cases. These cases have been arranged in order of increasing discrimination between classes.

We note here only one of other additional methods for obtaining S . From [4] we have that the expected number N_p of customers in the system of queues from group p is

$$N_p = \frac{\rho_p}{g_p(1-\rho)} \sum_{i=1}^P g_i \rho_i. \quad (32)$$

Also, for a job of average length $(1/\mu_p(1-\phi_p))$, we spend $\phi_p/\mu_p(1-\phi_p)$ seconds swapping. Thus we must have

$$S_p = \frac{\phi_p \rho_p \sum_{i=1}^P g_i \rho_i}{\mu_p(1-\phi_p)g_p(1-\rho)}. \quad (33)$$

Since $S = \sum_{p=1}^P S_p$, we sum and obtain (30).

Example 4: For our last example, we consider the finite-population case with M consoles, exponential service with mean $1/\mu$ and exponentially distributed think-time with mean $1/\gamma$ as studied in [1], [7], and [10]. From the curves given by Adiri and Avi-Itzhak [1], we may approximate $T(\tau)$ (the average total response time) by a linear function of τ . We take this as the solution form for $T(\tau)$ in the processor-shared case ($Q \rightarrow 0$). Thus we have (for zero swap time, $\phi=0$)

$$T(\tau) \cong K\tau \quad (34)$$

where K is some constant. To solve for this constant, we use the well-known result for T =average (over τ) of $T(\tau)$ (see [7]).

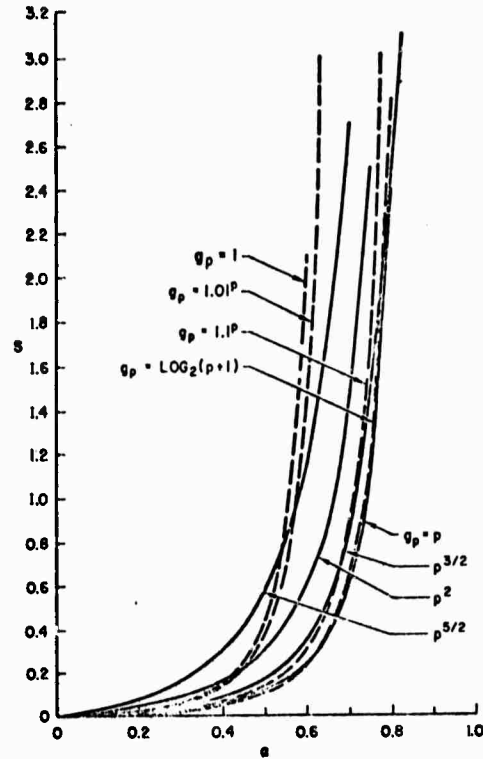


Fig. 5. Average swap time as a function of swap loss for the priority processor-shared system.

$$T = \frac{M/\mu}{1-\pi_0} - \frac{1}{\gamma} \quad (35)$$

where

$$\pi_0 = \left[\sum_{m=0}^M \frac{M!}{(M-m)!} \left(\frac{\gamma}{\mu} \right)^m \right]^{-1}. \quad (36)$$

But

$$T = \int_0^\infty T(\tau) dB(\tau) \quad (37)$$

where

$$B(\tau) = 1 - e^{-\mu\tau}.$$

From (34) and (37) we obtain

$$K = \mu T.$$

Thus, for the zero swap-time case

$$T(\tau) \cong \mu T \tau = \left[\frac{M}{1-\pi_0} - \frac{\mu}{\gamma} \right] \tau. \quad (38)$$

Now for $\phi > 0$, we merely use $\mu(1-\phi)$ rather than μ to obtain

$$T(\tau) \cong \left[\frac{M}{1-\pi_0} - \frac{\mu(1-\phi)}{\gamma} \right] \tau \quad (39)$$

where

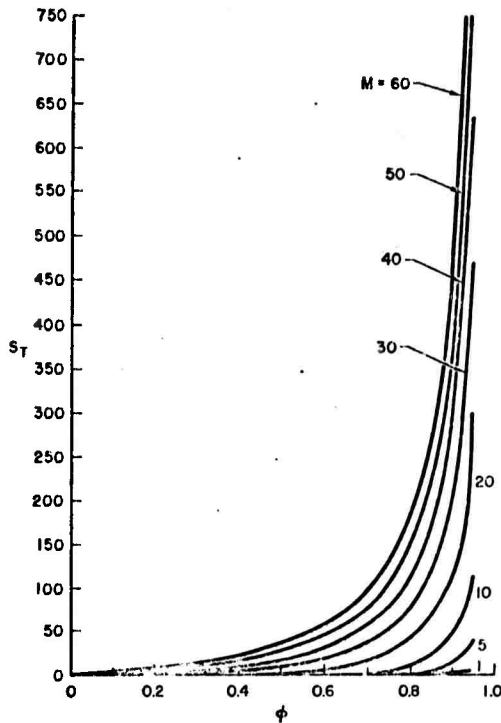


Fig. 6. Average swap time as a function of percentage swap time for the finite console processor-shared case.

$$\pi_0 = \left[\sum_{m=0}^M \frac{M!}{(M-m)!} \left[\frac{\gamma}{\mu(1-\phi)} \right]^m \right]^{-1}. \quad (40)$$

Applying Corollary 2 of Theorem 2, we obtain for the average system (queue plus service) swap time S_T ,

$$S_T \cong \lambda \phi \int_0^{\infty} \tau e^{-\mu(1-\phi)\tau} \mu(1-\phi) T d\tau$$

where λ , the average input and output rate, is clearly $\lambda = \mu(1-\phi)(1-\pi_0)$. Thus

$$S_T \cong (1 - \pi_0) \phi T'$$

$$S_T \cong \phi \left[\frac{M}{\mu(1-\phi)} - \frac{(1-\pi_0)}{\gamma} \right]$$

where π_0 is given by (40). S_T is plotted versus ϕ for various M in Fig. 6 for $1/\mu = 0.88$ and $1/\gamma = 35.2$.

CONCLUSION

We have shown how to solve for the expected swap time expended on all customers in the system of queues. This we have done for the time-sharing systems ($Q > 0$) in Theorem 1 and for the processor-sharing systems ($Q \rightarrow 0$) in Theorem 2. The examples given show the ease of obtaining results for the processor-sharing case.

REFERENCES

- [1] I. Adiri and B. Avi-Itzhak, "A time-sharing queue with a finite number of customers," *J. ACM*, vol. 16, no. 2, pp. 315-323, April 1969.
- [2] E. G. Coffman and L. Kleinrock, "Feedback queueing models for time-shared systems," *J. ACM*, vol. 15, no. 4, pp. 549-576, October 1968.
- [3] L. Kleinrock, "On swap time in time-shared systems," *1969 Proc. IEEE Computer Group Conf.* (Minneapolis, Minn.), pp. 37-41.
- [4] —, "Time-shared systems: A theoretical treatment," *J. ACM*, vol. 14, no. 2, pp. 242-261, April 1967.
- [5] —, "Analysis of a time-shared processor," *Naval Res. Logistics Quart.*, vol. 11, no. 10, pp. 59-73, March 1964.
- [6] L. Kleinrock and E. G. Coffman, "Distribution of attained service in time-shared systems," *J. Computer Sys. Sci.*, vol. 1, no. 3, pp. 287-298, October 1967.
- [7] L. Kleinrock, "Certain analytic results for time-shared processors," *1968 Proc. IFIP Cong.* (Edinburgh, Scotland), pp. D119-D125.
- [8] B. Krishnamoorthi and R. C. Wood, "Time-shared computer operations with both interarrival and service times exponential," *J. ACM*, vol. 13, no. 3, pp. 317-338, July 1966.
- [9] J. M. McKinney, "A survey of analytical time-sharing models," *Computing Surveys*, vol. 1, no. 2, pp. 105-116, June 1969.
- [10] A. L. Scherr, *An Analysis of Time-Shared Computer Systems*. Cambridge, Mass.: M.I.T. Press, 1967.
- [11] L. E. Schrage, "The queue M/G/1 with feedback to lower priority queues," *Management Sci.*, ser. A., vol. 13, pp. 466-474, 1967.

APPENDIX B

A CONTINUUM OF TIME-SHARING SCHEDULING ALGORITHMS

by

Leonard Kleinrock

A continuum of time-sharing scheduling algorithms*

by LEONARD KLEINROCK

University of California
Los Angeles, California

INTRODUCTION

The study of time-sharing scheduling algorithms has now reached a certain maturity. One need merely look at a recent survey by McKinney¹ in which he traces the field from the first published paper in 1964² to a succession of many papers during these past six years. Research which is currently taking place within the field is of the nature whereby many of the important theoretical questions will be sufficiently well answered in the very near future so as to question the justification for continuing extensive research much longer without first studying the overall system behavior.

Among the scheduling algorithms which have been studied in the past are included the round robin (RR), the feedback model with N levels (FB _{N}), and variations of these.¹ The models introduced for these scheduling algorithms gave the designer some freedom in adjusting system performance as a function of service time but did not range over a continuum of system behaviors. In this paper we proceed in that direction by defining a model which allows one to range from the first come first served algorithm all the way through to a round robin scheduling algorithm. We also find a variety of other models within a given family which have yet to be analyzed.

Thus the model analyzed in this paper provides to the designer a degree of freedom whereby he may adjust the relative behavior for jobs as a function of service time; in the past such a parameter was not available. Moreover, the method for providing this adjustment is rather straightforward to implement and is very easily changed by altering a constant within the scheduler.

* This work was supported by the Advanced Research Projects Agency of the Department of Defense (DAHC15-69-C-0285).

A GENERALIZED MODEL

In an earlier paper³ we analyzed a priority queueing system in which an entering customer from a particular priority group was assigned a zero value for priority but then began to increase in priority linearly with time at a rate indicative of his priority group. Such a model may be used for describing a large class of time-sharing scheduling algorithms. Consider Figure 1. This figure defines the class of scheduling algorithms which we shall consider. The principle behind this class of algorithms is that when a customer is in the system waiting for service then his priority (a numerical function) increases from zero (upon his entry) at a rate α ; similarly, when he is in service (typically with other customers sharing the service facility simultaneously with him as in a processor shared system⁴) his priority changes at a rate β . All customers possess the same parameters α and β . Figure 1 shows the case where both α and β are positive although, as we shall see below, this need not be the case in general. The history of a customer's priority value then would typically be as shown in Figure 1 where he enters the system at time t_0 with a 0 value of priority and begins to gain priority at a rate α . At time t_1 he joins those in service after having reached a value of priority equal to $\alpha(t_1 - t_0)$. When he joins those in service he shares on an equal basis the capacity of the service facility and then continues to gain priority at a different rate, β . It may be that a customer is removed from service before his requirement is filled (as may occur when one of the slopes is negative); in this case, his priority then grows at a rate of α again, etc. At all times, the server serves *all* those with the highest value of priority. Thus we can define a slope for priority while a customer is queueing and another slope for priority while a cus-

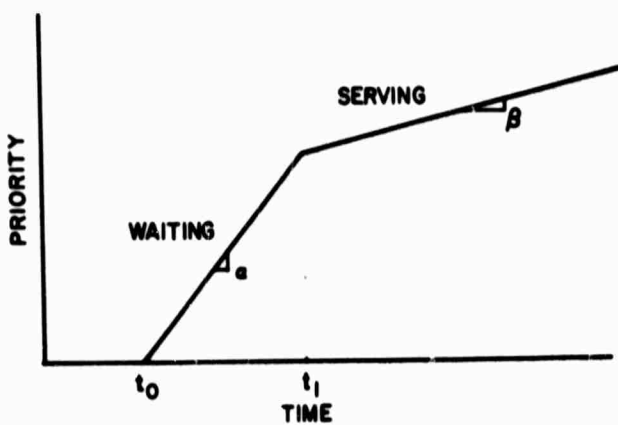


Figure 1—Behavior of the time-varying priority

customer is being served as

$$\text{queueing slope} = \alpha \quad (1)$$

$$\text{serving slope} = \beta. \quad (2)$$

A variety of different kinds of scheduling algorithms follow from this model depending upon the relative values of α and β . For example, when both α and β are positive and when $\beta \geq \alpha$ then it is clear that customers in the queue can never catch up to the customer in service since he is escaping from the queueing customers at least as fast as they are catching up to him; only when the customer in service departs from service after his completion will another customer be taken into service. This new customer to be taken into the service facility is that one which has the highest value of priority. Thus we see that for the range

$$0 < \alpha \leq \beta \quad (3)$$

we have a pure *first come first served* (FCFS) scheduling algorithm. This is indicated in Figure 2 where we show the entire structure of the general model.

Now consider the case in which

$$0 \leq \beta \leq \alpha. \quad (4)$$

This is the case depicted in Figure 1. Here we see that the group of customers being served (which act among themselves in a processor-shared round robin (RR) fashion) is attempting to escape from the group of customers in the queue; their attempt is futile, however, and it is clear from this range of parameters that the queueing customers will eventually each catch up with the group being served. Thus the group being served is selfishly attempting to maintain the service capacity for themselves alone and for this reason we refer to this system as the *selfish round robin* (SRR).

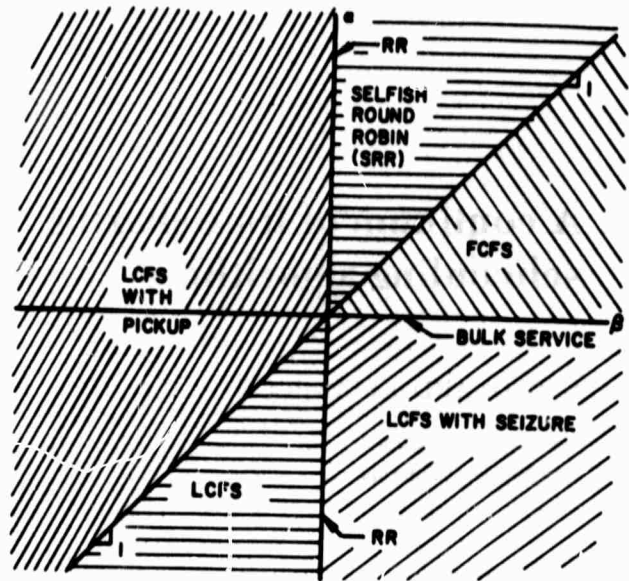


Figure 2—The structure of the general model

What happens in this case is that entering customers spend a period of time in the queue and after catching up with the serving group proceed to be served in a round robin fashion. The duration of the time they spend in the queue depends upon the relative parameters α and β as we shall see below. It is clear however that for $\beta = 0$ we have the case that customers in service gain no priority at all. Thus any newly entering customer would have a value of priority exactly equal to that of the group in service and so will immediately pass into the service group. Since all serving customers share equally, we see that the limiting case, $\beta = 0$, is a processor-sharing round robin (RR) scheduling algorithm! It happens that SRR yields to analysis very nicely (whereas some of the other systems mentioned below are as yet unsolved) and the results of this analysis are given in the next section.

Another interesting range to consider is that for which

$$\alpha \leq \beta < 0. \quad (5)$$

Here we have the situation in which queueing customers lose priority faster than serving customers do; in both cases however, priority decreases with time and so any newly entering customer will clearly have the highest priority and will take over the complete service facility for themselves. This most recent customer will continue to occupy the service facility until either he leaves due to a service completion or some new customer enters the system and ejects him. Clearly what we have here is a classical *last come first served* (LCFS) scheduling algorithm as is indicated in Figure 2.

Now consider the range

$$\alpha < 0 < \beta. \quad (6)$$

In this case a waiting customer loses priority whereas a customer in service gains priority. When an arriving customer finds a customer in service who has a negative value for priority then this new customer preempts the old customer and begins service while at the same time his priority proceeds to increase at a rate β ; from here on no other customer can catch him and this customer will be served until completion. Upon his completion, service will then revert back to that customer with the largest value of priority. Since customers lose priority with queueing time, then all customers in the system when our lucky customer departed must have negative priority. One of these will be chosen and will begin to gain priority; if now he is lucky enough to achieve a positive priority during his service time, then he will seize the service facility and maintain possession until his completion. Thus we call this range *LCFS with seizure* (see Figure 2).

In the special case

$$\alpha = 0 < \beta \quad (7)$$

we have the situation in which a newly emptied service facility will find a collection of customers who have been waiting for service and who have been kept at a zero value priority. Since all of these have equal priority they will all be taken into service simultaneously and then will begin to gain priority at a rate $\beta > 0$. Any customers arriving thereafter must now queue in bulk fashion since they cannot catch up with the current group in service. Only when that group finishes service completely will the newly waiting group be taken into service. We refer to this case as *bulk service*.

The last case to consider is in the range

$$\beta < 0, \quad \beta < \alpha. \quad (8)$$

In this case a customer being served always loses priority whereas a queueing customer loses priority at a slower rate or may in fact gain priority. Consequently, serving customers will tend to "run into" queueing customers and pick them up into the service facility at which point the entire group continues to decrease in priority at rate β . We refer to this region as *LCFS with pickup* (see Figure 2).

Thus Figure 2 summarizes the range of scheduling algorithms which this two-parameter priority function can provide for us. We have described a number of regions of interest for this class of algorithms. The FCFS, LCFS, and RR systems, of course, are well known and solved. The three regions given by Equations 4, 6, and 8 are as yet unsolved. As mentioned

before, the SRR system yields very nicely to analysis and that analysis is given in this paper. This system has the interesting property that we may vary its parameters and pass smoothly from the FCFS system through the SRR class to the familiar RR system. The others (LCFS with seizure and LCFS with pickup) are as yet unsolved and appear to be more difficult to solve than the SRR. Of course other generalizations to this scheme are possible, but these too are yet to be studied. Among these generalizations, for example, is the case where each customer need not have the same α and β ; also one might consider the case where growth (or decay) of priority is a non-linear function of time. Of all these cases we repeat again that the SRR has been the simplest to study and its analysis follows in the next section.

THE SELFISH ROUND ROBIN (SRR) SCHEDULING ALGORITHM

We consider the system for which customers in service gain priority at a rate less than or equal to the rate at which they gained priority while queueing (see Equation (4)); in both cases the rate of gain is positive. We assume that the arrival process is Poisson at an average rate of λ customers per second

$$P[\text{inter-arrival time} \leq t] = 1 - e^{-\lambda t} \geq 0 \quad (9)$$

and that the service times are exponentially distributed

$$P[\text{service time} \leq x] = 1 - e^{-\mu x} \quad x \geq 0 \quad (10)$$

Thus the two additional parameters of our system are

$$\text{average arrival rate} = \lambda \quad (11)$$

$$\text{average service time} = 1/\mu \quad (12)$$

As usual, we define the utilization factor

$$\rho = \lambda/\mu \quad (13)$$

For the range of α, β under consideration it is clear that once a customer enters the service facility he will not leave until his service is complete. Consequently, we may consider the system as broken into two parts: first, a collection of queued customers; and second, a collection of customers in service. Figure 3 depicts this situation where we define*

$$T_w = E[\text{time spent in queue box}] \quad (14)$$

$$T_s = E[\text{time spent in service box}] \quad (15)$$

$$N_w = E[\text{number in queue box}] \quad (16)$$

$$N_s = E[\text{number in service box}] \quad (17)$$

* The notation $E[x]$ reads as "the expectation of x ."

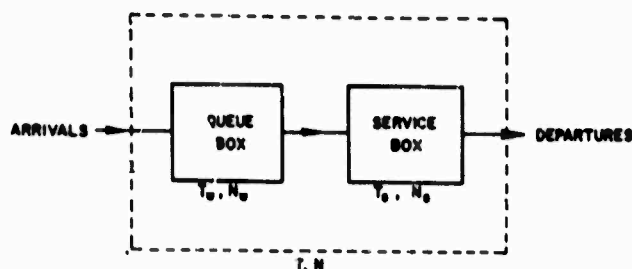


Figure 3—Decomposition of the SRR system

We further define

$$T = T_w + T_s = E[\text{time in system}] \quad (18)$$

$$N = N_w + N_s = E[\text{number in system}] \quad (19)$$

Due to the memoryless property of the exponential service time distribution, it is clear that the average number in system and average time in system are independent of the order of service of customers; this follows both from intuition and from the conservation law given in Reference 5. Thus we have immediately

$$T = \frac{1/\mu}{1 - \rho} \quad (20)$$

$$N = \frac{\rho}{1 - \rho} \quad (21)$$

For our purposes we are interested in solving for the average response time for a customer requiring t seconds of service; that is for a customer requiring t seconds of complete attention by the server or $2t$ seconds of service from the server when he is shared between two customers, etc. Recall that more than one customer may simultaneously be sharing the attention of the service facility and this is just another class of processor-sharing systems.⁴ Thus our goal is to solve for

$$T(t) = E[\text{response time for customer requiring } t \text{ seconds of service}] \quad (22)$$

where by response time we mean total time spent in the system. The average of this conditional response time without regard to service time requirement is given by Equation 20. Due to our decomposition we can write immediately

$$T(t) = T_w(t) + T_s(t) \quad (23)$$

where $T_w(t)$ is the expected time spent in the queue box for customers requiring t seconds of service and $T_s(t)$ is the expected time spent in the service box for customers requiring t seconds of service. Since the

system is unaware of the customer's service time until he departs from the system, it is clear that the time he spends in the queue box must be independent of this service time and therefore

$$T_w(t) = T_w \quad (24)$$

Let us now solve for $T_s(t)$. We make this calculation by following a customer, whom we shall refer to as the "tagged" customer, through the system given that this customer requires t seconds of service. His time in the queue box will be given by Equation 24. We now assume that this tagged customer has just entered the service box and we wish to calculate the expected time he spends there. This calculation may be made by appealing to an earlier result. In Reference 4, we studied the case of the processor-shared round robin system (both with and without priorities). Theorem 4 of that paper gives the expected response time conditioned on service time and we may use that result here since the system we are considering, the service box, appears like a round robin system. However, the arrival rate of customers to the service box conditioned on the presence of a tagged customer in that box is no longer λ , but rather some new average arrival rate λ' . In order to calculate λ' we refer the reader to Figure 4. In this figure we show that two successive customers arrive at times t_1 and t_2 where the average time between these arrivals is clearly $1/\lambda$. The service group moves away from the new arrivals at a rate β and the new arrivals chase the service group at a rate α ; as shown in Figure 4, these two adjacent arrivals catch up with the service group where the time between their arrival to the service box is given by $1/\lambda'$. Recall that the calculation we are making is conditioned on the fact that our tagged customer remains in the service box during the interval of interest; therefore the service box is guaranteed not to empty over the period of our

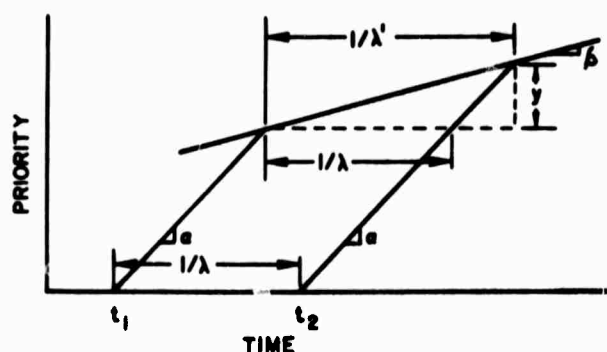


Figure 4—Calculation of the conditional arrival rate to the service box

calculations. λ' is easily calculated by recognizing that the vertical offset y may be written in the following two ways

$$y = \left(\frac{1}{\lambda'}\right)\beta$$

$$y = \left(\frac{1}{\lambda'} - \frac{1}{\lambda}\right)\alpha$$

and so we may solve for λ' as follows

$$\lambda' = \lambda \left(1 - \frac{\beta}{\alpha}\right) \quad (25)$$

(recall that for the SRR system $\beta \leq \alpha$). For convenience we now define

$$\rho' = \lambda'/\mu \quad (26)$$

We may now apply Theorem 4 of Reference 4 and obtain the quantity we are seeking, namely,

$$T_s(t) = \frac{t}{1 - \rho'} \quad (27)$$

The only difference between Equation 27 and the referenced theorem is that here we use ρ' instead of ρ since in all cases we must use the appropriate utilization factor for the system under consideration. That theorem also gives us immediately that

$$N_s = \frac{\rho'}{1 - \rho'} \quad (28)$$

This last equation could be derived from Equation 27 and the application of Little's result⁴ which states that

$$\lambda' T_s = N_s \quad (29)$$

and where

$$T_s = \int_0^{\infty} T_s(t) \mu e^{-\mu t} dt$$

$$T_s = \frac{1/\mu}{1 - \rho'} \quad (30)$$

We may now substitute Equation 27 into Equation 23 to give

$$T(t) = T_w + \frac{t}{1 - \rho'} \quad (31)$$

In order to evaluate T_w we form the average with respect to t over both sides of Equation 31 to obtain

$$\int_0^{\infty} T(t) \mu e^{-\mu t} dt = T_w + \int_0^{\infty} \frac{t}{1 - \rho'} \mu e^{-\mu t} dt$$

and so

$$T = T_w + \frac{1/\mu}{1 - \rho'} \quad (32)$$

Using Equation 20 we have the result

$$T_w = \frac{1/\mu}{1 - \rho} - \frac{1/\mu}{1 - \rho'} \quad (33)$$

Upon substituting Equation 33 into Equation 31 we obtain our final result as

$$T(t) = \frac{1/\mu}{1 - \rho} + \frac{t - 1/\mu}{1 - \rho'} \quad (34)$$

Another convenient form in which to express this result is to consider the average time wasted in this SRR system where wasted time is any extra time a customer spends in the system due to the fact that he is sharing the system with other customers. Thus, by definition, we have

$$W(t) = T(t) - t \quad (35)$$

and this results in

$$W(t) = \frac{\rho/\mu}{1 - \rho} + \frac{(t - 1/\mu)\rho'}{1 - \rho'} \quad (36)$$

In both Equations 34 and 36 we observe for the case of a customer whose service time is equal to the average service time ($1/\mu$) that his average response time and average wasted time are the same that he would encounter for any SRR system; thus his performance is the same that he would receive, for example, in a FCFS system. We had observed that correspondence between the RR system and the FCFS system in the past; here we show that it holds for the entire class of SRR systems. In Figure 5 below we plot the performance of the class of SRR systems by showing the dependence of the wasted time for a customer whose service time is t seconds as a function of his service time. We show this for the case $\rho = 3/4$ and $\mu = 1$. The truly significant part regarding the behavior of the SRR system is that the dependence of the conditional response time upon the service time is linear. Once observed, this result is intuitively pleasing if we refer back to Figure 3. Clearly, the time spent in the queue box is some constant independent of service time. However, the time spent in the service box is time spent in a round robin system since all customers in that box share equally the capability of the server; we know that the response time for the round robin system is directly proportional to service time required (in fact, as shown in Reference 8, this statement is true even for arbitrary service time). Thus the total time spent in the SRR system must be equal to some con-

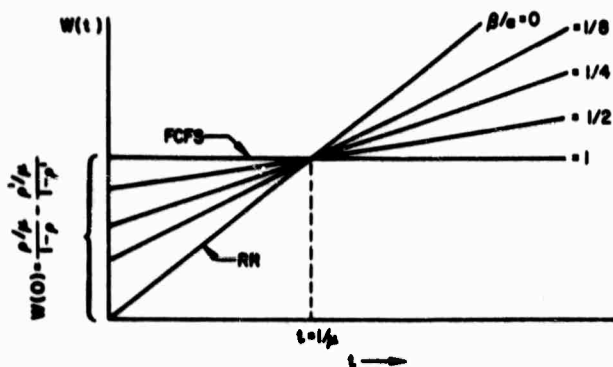


Figure 5—Performance of the SRR system

stant plus a second term proportional to service time as in fact our result in Equation 34 indicates. Again we emphasize the fact that customers whose service time requirements are greater than the average service time requirement are discriminated against in the SRR system as compared to a FCFS system; conversely, customers whose service time requirement is less than the average are treated preferentially in the SRR system and compared to the FCFS system. The degree of this preferential treatment is controlled by the parameters α and β giving the performance shown in Figure 5.

CONCLUSION

In this paper we have defined a continuum of scheduling algorithms for time-shared systems by the introduction of two new parameters, α and β . The class so defined is rather broad and its range is shown in Figure 2. We have presented the analysis for the range of parameters that is given in Equation 4 and refer to this new system as the selfish round robin (SRR) scheduling algorithm. Equation 34 gives our result for the average response time conditioned on the required service time and we observed that this result took the especially simple form of a constant plus a term linearly dependent upon the service time. Moreover, we observe that the parameters α and β appear in the solution only as the ratio β/α . This last is not overly surprising since a similar observation was made in the paper³ which was our point of departure for the model described herein; namely, there too the slope parameters

appeared only as ratios. Thus in effect we have introduced one additional parameter, the ratio β/α , and it is through the use of this parameter that the designer of a time-sharing scheduling algorithm is provided a degree of freedom for adjusting the extent of discrimination based upon service time requirements which he wishes to introduce into his algorithm; the implementation of this degree of freedom is especially simple. The range of the algorithm is from the case where there is zero discrimination based on service time, namely the FCFS system, to a case where there is a strong degree of discrimination, namely the RR system.

The mathematical simplicity of the SRR algorithm is especially appealing. Nevertheless, the unsolved systems referred to in this paper should be analyzed since they provide behavior distinct from the SRR. In any event, this continuum of algorithms is simply implemented in terms of the linear parameters α and β , and the scheduling algorithm can easily choose the desired behavior by adjusting α and β appropriately.

REFERENCES

- 1 J M MCKINNEY
A survey of analytical time-sharing models
Computing Surveys Vol 1 No 2 pp 105-116 June 1969
- 2 L KLEINROCK
Analysis of a time-shared processor
Naval Research Logistics Quarterly Vol 11 No 1 pp 59-73 March 1964
- 3 L KLEINROCK
A delay dependent queue discipline
Naval Research Logistics Quarterly Vol 11 No 4 pp 329-341 December 1964
- 4 L KLEINROCK
Time-shared systems: A theoretical treatment
JACM Vol 14 No 2 pp 242-261 April 1967
- 5 L KLEINROCK
A conservation law for a wide class of queueing disciplines
Naval Research Logistics Quarterly Vol 12 No 2 pp 181-192 June 1965
- 6 J D C LITTLE
A proof of the queueing formula $L = \lambda W$
Operations Research Vol 9 pp 383-387 1961
- 7 L KLEINROCK
Distribution of attained service in time-shared systems
J of Computers and Systems Science Vol 3 pp 287-298 October 1967
- 8 M SAKATA S NOGUCHI J OIZUMI
Analysis of a processor shared queueing model for time-sharing systems
Proc of the Second Hawaii International Conference on Systems Science pp 625-628
University of Hawaii Honolulu Hawaii January 22-24 1969

APPENDIX C

MULTILEVEL PROCESSOR-SHARING QUEUEING MODELS
FOR TIME-SHARED SYSTEMS

by

L. Kleinrock and R. R. Muntz

BLANK PAGE

MULTILEVEL PROCESSOR-SHARING QUEUEING MODELS FOR TIME-SHARED SYSTEMS*

L. Kleinrock and R. R. Muntz

University of California

Los Angeles, California USA

ABSTRACT

Scheduling algorithms for time-shared computing facilities are considered in terms of a queueing theory model. The extremely useful limit of "processor-sharing" is adopted, wherein the quantum of service shrinks to zero; this approach greatly simplifies the problem. A class of algorithms is studied for which the scheduling discipline may change for a given job as a function of the amount of service received by that job. These multilevel disciplines form a natural extension to many of the disciplines previously considered.

Solved for is the average response time for jobs conditioned on their service requirement. Explicit solutions are given for the system $M/G/1$ in which levels may be first-come-first-served (FCFS) or feedback (FB) in any order; in addition, the round-robin (RR) may be used at the first level. An integral equation is developed which defines (but does not as yet provide a solution for) the RR case at arbitrary level. The special case in which RR is used at the last level is solved under the condition that the service time behave exponentially for this last level.

Examples are described for which the average response time is plotted. These examples display the great versatility of the results and demonstrate the flexibility available for the intelligent design of discriminatory treatment among jobs (in favor of short jobs and against long jobs) in time-shared system design.

1. INTRODUCTION

Queueing models have been used successfully in the analysis of time-shared computer systems since the appearance of the first applied paper in this field in 1964 [1]. The recent survey [2] of such analytical work attests to this fact. One of the first papers to consider the effect of feedback in queueing systems was due to Takacs [3].

Generally, an arrival enters the time-shared system and competes for the attention of a single processing unit. This arrival is forced to wait in a system of queues until he is permitted a quantum of service time; when this quantum expires, he is then required to join the system of queues to await his second quantum, etc. The precise model for the system is developed in Section 2. In 1967 the notion of allowing the quantum to shrink to zero was first studied [4] and is referred to as "processor-sharing." As the name implies, this zero-quantum limit provides a share or portion of the processing unit to many customers simultaneously; in the case of round-robin (RR) scheduling [4], all customers in the system simultaneously share (equally or on a priority basis) the processor, whereas in the feedback (FB) scheduling [5] only that set of customers with the smallest attained service share the processor. We use the term processor-sharing here since it is the processing unit itself (the central processing unit of the computer) which is being shared among the set of the customers; the phrase "time-sharing" will be reserved to imply that customers are waiting sequentially for their turn to use the entire processor for a finite quantum. In studying the literature one finds that the obtained results appear in a rather complex form and this complexity arises from the fact that the quantum is typically assumed to be finite as opposed to infinitesimal. When one allows the quantum to shrink to zero, giving rise to a processor-sharing system, then the difficulty in analysis as well as in the form of results disappears in large part; one is thus encouraged to consider only the processor-sharing case. Clearly, this limit of infinitesimal quantum is an ideal and can seldom be reached in practice due to considerations of swap time; nevertheless, its extreme simplicity in analysis and results brings us to the studies reported in this paper.

The two processor-sharing systems studied in the past are the RR and the FB [4,5]. Typically, the quantity solved for is the expected response time conditioned on the customer's service time; response time is the elapsed time from when a customer enters the system until he leaves completely serviced. This measure is especially important since it exposes the preferential treatment given to short jobs at the expense of the long jobs. Clearly, this discrimination is purposeful since it is the desire in time-shared systems that small requests should be allowed to pass through the system quickly. In 1969 the distribution for the response time in the RR system was found [6]. In this paper we consider the case of mixed scheduling algorithms whereby customers are treated according to the RR algorithm, the FB algorithm, or first come first served (FCFS) algorithm, depending upon how much total service time they have already received. Thus, as a customer proceeds through the system obtaining service at various rates he is treated according to different disciplines; the policy which is applied among customers in different levels is that of the FB system as explained further in Section 2. This natural generalization of the previously studied processor-sharing systems allows one to create a large number of new and interesting disciplines whose solutions we present.

*This work was supported by the Advanced Research Projects Agency of the Department of Defense (DARC-15-69-C-0285).

2. THE MODEL

The model we choose to use in representing the scheduling algorithms is drawn from queueing theory. This corresponds to the many previous models studied [1,2,4,5,6,7,8], all of which may be thought of in terms of the structure shown in Fig. 2.1. In this figure we indicate that new requests enter the system of queues upon arrival. Whenever the computer's central processing unit (CPU) becomes free, some customer is allowed into the service facility for an amount of time referred to as a quantum. If during this quantum, the total accumulated service for a customer equals his required service time, then he departs the system; if not, at the end of his quantum, he cycles back to the system of queues and waits until he is next chosen for additional service. The system of queues may order the customers according to a variety of different criteria in order to select the next customer to receive a quantum. In this paper, we assume that the only measure used in evaluating this criterion is the amount of attained service (total service so far received).

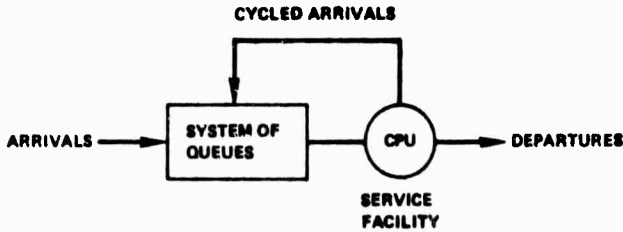


Figure 2.1. The Feedback Queueing Model

In order to specify the scheduling algorithm in terms of this model, it is required that we identify the following:

a. The customer interarrival time distribution. We assume this to be exponential, i.e.,

$$P[\text{interarrival time} \leq t] = 1 - e^{-\lambda t} \quad t \geq 0 \quad (2.1)$$

where λ is the average arrival rate of customers.

b. The distribution of required service time in the CPU. This we assume to be arbitrary (but independent of the interarrival times). We thus assume

$$P[\text{service time} \leq x] = B(x) \quad (2.2)$$

Also assume $1/\mu$ = average service time

c. The quantum size. Here we assume a processor-shared model in which customers receive an equal but vanishingly small amount of service each time they are allowed into service. For more discussion of such systems, see [4,6,7].

d. The system of queues. We consider here a generalization and consolidation of many systems studied in the past. In particular, we define a set of attained service times $\{a_i\}$ such that

$$0 = a_0 < a_1 < a_2 < \dots < a_N < a_{N+1} = \infty \quad (2.3)$$

The discipline followed for a job when it has attained service, τ , in the interval

$$a_{i-1} \leq \tau \leq a_i \quad i = 1, 2, \dots, N+1 \quad (2.4)$$

will be denoted as D_i . We consider D_i for any given level i to be either: FIRST COME FIRST SERVED (FCFS); PROCESSOR SHARED-FB (FB); or ROUND-ROBIN PROCESSOR SHARED (RR). The FCFS system needs no explanation. The FB system gives service next to that customer who so far has least attained service; if there is a tie (among K customers, say) for this position, then all K members in the tie get served simultaneously (each attaining useful service at a rate of $1/K$ sec/sec), this being the nature of processor sharing systems. The RR processor sharing system shares the service facility among all customers present (say J customers)

giving attained service to each at a rate of $1/J$ sec/sec. Moreover, between intervals, the jobs are treated as a set of FB disciplines. See Fig. 2.2. For example, for $N=0$, we have the usual single-level case of either FCFS, RR or FB.

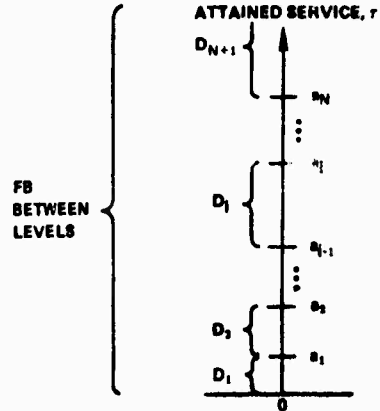


Figure 2.2. Intervals of Attained Service, with Disciplines, D_i

For $N=1$, we could have any of nine disciplines (FCFS followed by FCFS, ..., RR followed by RR); note that FB followed by FB is just a single FB system (due to the overall FB policy between levels).

As an illustrative example, consider the $N=2$ case shown in Fig. 2.3.

Any new arrivals begin to share the processor in a RR fashion with all other customers who so far have less than 2 seconds of attained service. Customers in the range of $2 < \tau < 6$ may get served only if no customers present have had less than 2 seconds of service; in such a case, that customer (or customers) with the least attained service will proceed to occupy the service in an FB fashion until they either leave, or reach $\tau = 6$, or some new customer arrives (in which case the overall FB rule provides that the RR policy at level 1 preempts). If all customers have $\tau > 6$, then the "oldest" customer will be served to completion unless interrupted by a new arrival. The history of some customers in this example system is shown in Fig. 2.4. We denote customer n by C_n . Note that the slope of attained

Figure 2.3. Example of $N=2$

service varies as the number of customers simultaneously being serviced changes. We see that C_2 required 5 seconds of service and spent 14 seconds in system (i.e., response time of 14 seconds).

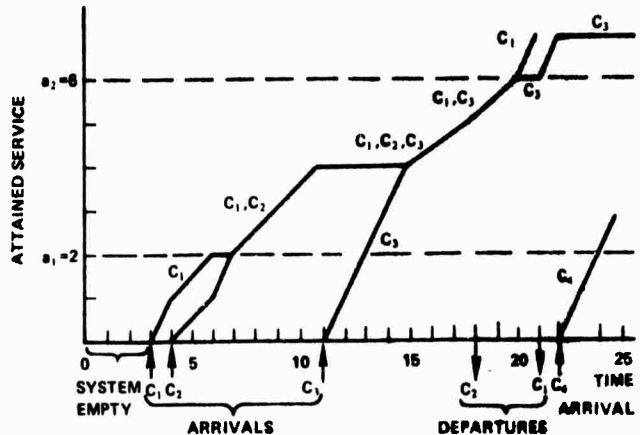


Figure 2.4. History of Customers in Example

So much for the system specification. We may summarize by saying that we have an M/G/1 queueing system* model with processor sharing and with a generalized multilevel scheduling structure.

The quantity we wish to solve for is

$$T(t) = E[\text{response time for a customer requiring a total of } t \text{ seconds of attained service}] \quad (2.5)$$

We further make the following definitions:

$$T_i(t) = E[\text{time spent in interval } i \text{ (} a_{i-1}, a_i \text{) for customers requiring a total of } t \text{ seconds of attained service}] \quad (2.6)$$

We note that

$$T_i(t) = T_i(t') \quad \text{for } t, t' > a_i \quad (2.7)$$

Furthermore, we have, for $a_{k-1} < t \leq a_k$, that

$$T(t) = \sum_{i=1}^k T_i(t) \quad (2.8)$$

Also, we find it convenient to define the following quantities with respect to $B(t)$ truncated at $t = x$:

$$E_{<x} = \int_0^x t dB(t) + x \int_x^\infty dB(t) \quad (2.9)$$

$$\overline{t}_{<x}^2 = \int_0^x t^2 dB(t) + x^2 \int_x^\infty dB(t) \quad (2.10)$$

$$\rho_{<x} = \lambda E_{<x} \quad (2.11)$$

$$W_x = \frac{\lambda \overline{t}_{<x}^2}{2(1 - \rho_{<x})} \quad (2.12)$$

Note that W_x represents the expected work found by a new arrival in the system M/G/1 where the service time distribution is $B(t)$ truncated at x .

3. RESULTS FOR MULTILEVEL QUEUEING SYSTEMS

We wish to find an expression for $T(t)$, the mean system time (i.e., average response time) for jobs with service time t such that $a_{i-1} < t \leq a_i$, i.e., jobs which reach the i^{th} level queue and then leave the system. To accomplish this it is convenient to isolate the i^{th} level to some extent. We make use of the following two facts.

1. By the assumption of preemptive priority of lower level queues (i.e., FB discipline between levels) it is clear that jobs in levels higher than the i^{th} level can be ignored. This follows since these jobs cannot interfere with the servicing of the lower levels.
2. We are interested in jobs that will reach the i^{th} level queue and then depart from the system before passing to the $(i+1)^{\text{st}}$ level. The system time of such a job can be thought of as occurring in two parts. The first portion is the time from the job's arrival to the queueing system until the group at the i^{th} level is serviced for the first time after this job has reached the i^{th} level. The second portion starts with

the end of the first portion and ends when the job leaves the system. It is easy to see that both the first and second portions of the job's system time are unaffected by the service disciplines used in levels 1 through $i-1$. Therefore, we can assume any convenient disciplines. In fact, all these levels can be lumped into one equivalent level which services jobs with attained service between 0 and a_{i-1} seconds using any service discipline.

From (1) and (2) above it follows that we can solve for $T(t)$ for jobs that leave the system from the i^{th} level by considering a two-level system. The lower level services jobs with attained service between 0 and a_{i-1} whereas the second level services jobs with attained service between a_{i-1} and a_i . Jobs that would have passed to the $i+1^{\text{st}}$ level after receiving a_i seconds of service in the original system are now assumed to leave the system at that point. In other words the service time distribution is truncated at a_i .

3.1. i^{th} Level Discipline is FB

Consider the two-level system with the second level corresponding to the i^{th} level of the original system. Since we are free to choose the discipline used in the lower level, we can assume that the FB discipline is used in this level as well. Clearly the two-level system behaves like a single level FB system with service time distribution truncated at a_i . The solution for such a system is known [5,11]:

$$T(t) = \frac{t}{1 - \rho_{<t}} + \frac{\lambda \overline{t}_{<t}^2}{2(1 - \rho_{<t})^2} \quad (3.1)$$

3.2. i^{th} Level Discipline is FCFS

Consider again the two-level system with breakpoints at a_{i-1} and a_i . Regardless of the discipline in the lower level, a tagged job entering the system will be delayed by the sum of the work currently in both levels plus any new arrivals to the lower level queue during the interval this job is in the system. These new arrivals form a Poisson process with parameter λ and their contribution to the delay is a random variable whose first and second moments are $E_{<a_i}$ and $\overline{t}_{<a_i}^2$ respectively. By delay cycle analysis [9] we have

$$T(t) = \frac{W_{a_i} + t}{1 - \rho_{<a_i}} \quad (3.2)$$

It is also possible to use these methods for solving last-come-first-served and random order of service at any level.

3.3. i^{th} Level Discipline is RR

Results to date allow explicit solutions for only two cases. (1) RR in the first level and (2) RR in the last level with the added restriction that once a job has reached this level the distribution of remaining service time is exponential. The analysis will be developed for the general case as far as possible before being restricted to special cases.

We start by considering the two-level system with breakpoints at a_{i-1} and a_i . Consider the busy periods of the lower level. During each such busy period there may be a number of jobs that pass to the higher level. We choose to consider these arrivals to the higher level as occurring at the end of the lower level busy period so that there is a bulk arrival to the higher level at this time. We also choose to temporarily delete these lower level busy periods from the time axis. In effect we create a virtual time axis telescoped to delete the lower level busy periods. Since the time from the end of one lower level busy period to the start of the next is exponentially distributed (Poisson arrivals!), the arrivals to the higher level appear

*M/G/1 denotes the single-server queueing system with Poisson arrivals and arbitrary service time distribution.

in virtual time to be bulk arrivals at instants generated from a Poisson process with parameter λ .

Consider a job that requires $t = a_{i-1} + \tau$ seconds of service ($0 < \tau \leq a_i - a_{i-1}$). Let α_1 be the mean real time the job spends in the system until its arrival (at the end of the lower level busy period) at the higher level queue. Let $\alpha_2(\tau)$ be the mean virtual time the job spends in the higher level queue.

α_1 can be calculated using delay cycle analysis. The initial delay is equal to the mean work the job finds in the lower level on arrival plus the a_{i-1} seconds of work that it contributes to the lower level. This initial delay is expanded by new jobs arriving at the lower level by a factor of $1/(1 - \rho_{a_{i-1}})$ (see [9]). Therefore

$$\alpha_1 = \frac{1}{1 - \rho_{a_{i-1}}} \{W_{a_{i-1}} + a_{i-1}\} \quad (3.3)$$

If $\alpha_2(\tau)$ is the mean virtual time the job spends in the higher level, we can easily convert this to the mean real time spent in this level. In the virtual time interval $\alpha_2(\tau)$ there are an average of $\lambda \alpha_2(\tau)$ lower level busy periods that have been ignored. Each of these has a mean length of $\frac{\bar{t}_{a_{i-1}}}{1 - \rho_{a_{i-1}}}$.

Therefore, the mean real time the job spends in the higher level is given by

$$\alpha_2(\tau) + \lambda \alpha_2(\tau) \frac{\bar{t}_{a_{i-1}}}{1 - \rho_{a_{i-1}}} = \frac{\alpha_2(\tau)}{1 - \rho_{a_{i-1}}} \quad (3.4)$$

Combining these results we see that a job requiring $t = a_{i-1} + \tau$ seconds of service has mean system time given by

$$T(a_{i-1} + \tau) = \frac{1}{1 - \rho_{a_{i-1}}} \{W_{a_{i-1}} + a_{i-1} + \alpha_2(\tau)\} \quad (3.5)$$

The only unknown quantity in this equation is $\alpha_2(\tau)$.

To solve for $\alpha_2(\tau)$ we must, in general, consider an M/G/1 system with bulk arrivals and RR processor sharing. The only exception is the case of RR at the first level which has only single arrivals. Since the higher level queues can be ignored, the solution in this exceptional case is the same as for a round-robin single level system with service time distribution truncated at a_1 . Therefore, from [8] we have for the first level

$$T(t) = \frac{t}{1 - \rho_{a_1}} \quad 0 \leq t \leq a_1 \quad (3.6)$$

Let us now consider the bulk arrival RR system in isolation in order to solve for the virtual time spent in the higher level queue, $\alpha_2(\tau)$ which we temporarily write as $\alpha(\tau)$.

3.3.1. The Bulk Arrival, RR Model

We approach this problem by first considering a discrete time system with quantum size $q > 0$. We assume that arrivals and departures take place only at times that are integral multiples of q . For small q any continuous distribution can be approximated. By letting q approach 0 equations for continuous time systems can be found.

Let $n(iq)$ = the mean number of jobs in the system with iq seconds of attained service. (3.7)

σ_i = the probability that a job which has received iq seconds of service will require more than $(i+1)q$ seconds of service. (3.8)

\bar{a} = the mean bulk size of arrivals. (3.9)

b = the mean number of arrivals with a tagged job. (3.10)

Since we intend to let q approach 0, the position of the tagged job with respect to the jobs that arrive in the same group is not important. We will assume for convenience that the tagged job is the last job in the group.

The mean time until the tagged job has received its first quantum of service is given by

$$T_1 = \sum_{j=0}^{\infty} n(jq)q + bq + q \quad (3.11)$$

In general, the mean time between the $(i-1)^{st}$ and i^{th} quantum of service to the tagged job is given by

$$T_i = \sum_{j=0}^{\infty} (n(jq)\sigma_j\sigma_{j+1} \dots \sigma_{j+i-2}q) + \sum_{j=1}^{i-1} (\lambda \bar{a} T_j \sigma_0 \sigma_1 \dots \sigma_{i-j-2}q) + q + b(\sigma_0 \sigma_1 \dots \sigma_{i-1}q) \quad (3.12)$$

The first term represents the time required by those jobs which were initially in the system and will still be there after the tagged job has received $i-1$ quanta of service. The second term is the contribution due to jobs that have arrived since the tagged job entered the system. The third term is due to the tagged job itself. The last term results from those jobs which arrived with the tagged job and require more than $i-1$ quanta of service.

Dividing both sides of Eq. (3.12) by q we get

$$\frac{T_i}{q} = \sum_{j=0}^{\infty} n(jq)\sigma_j\sigma_{j+1} \dots \sigma_{j+i-2} + \sum_{j=1}^{i-1} \lambda \bar{a} T_j \sigma_0 \sigma_1 \dots \sigma_{i-j-2} + 1 + b\sigma_0 \sigma_1 \dots \sigma_{i-1} \quad (3.13)$$

Let $iq = t$ and $jq = x$. Then as $q \rightarrow 0$:

$$\frac{T_i}{q} + \alpha'(t) \equiv \frac{d\alpha(t)}{dt} \quad (3.14)$$

$$\sigma_j\sigma_{j+1} \dots \sigma_{j+i-2} \rightarrow \frac{1 - B(x+t)}{1 - B(x)} \quad (3.15)$$

$$n(jq) \rightarrow n(x) \quad (3.16)$$

$$\sigma_0 \sigma_1 \dots \sigma_{i-j-2} \rightarrow 1 - B(t-x) \quad (3.17)$$

$$\sigma_0 \sigma_1 \dots \sigma_{i-1} \rightarrow 1 - B(t) \quad (3.18)$$

Therefore as $q \rightarrow 0$ Eq. (3.13) becomes

$$\alpha'(t) = \int_0^{\infty} n(x) \frac{1 - B(x+t)}{1 - B(x)} dx + \lambda \bar{a} \int_0^t \alpha'(x) [1 - B(t-x)] dx + 1 + b[1 - B(t)] \quad (3.19)$$

that

From Kleinrock and Coffman [7] we also have

$$n(x) = \lambda \bar{a} [1 - B(x)] a'(x) \quad (3.20)$$

Substituting for $n(x)$ we have

$$\begin{aligned} a'(t) &= \lambda \bar{a} \int_0^\infty a'(x) [1 - B(x+t)] dx \\ &\quad + \lambda \bar{a} \int_0^t a'(x) [1 - B(t-x)] dx \\ &\quad + 1 + b[1 - B(t)] \end{aligned} \quad (3.21)$$

This integral equation defines $a'(t)$ for the case of bulk arrival to a RR processor-sharing M/G/1 system. Unfortunately no general solution has been found for this equation in terms of $B(t)$. However, for exponential service time the equation can be solved.

3.3.1a. M/M/1 with Bulk Arrival. In this case

$$B(t) = 1 - e^{-\mu t} \quad (3.22)$$

Therefore the Eq. (3.21) becomes

$$\begin{aligned} a'(t) &= \lambda \bar{a} \int_0^\infty a'(x) e^{-\mu(x+t)} dx \\ &\quad + \lambda \bar{a} \int_0^t a'(x) e^{-\mu(t-x)} dx \\ &\quad + 1 + b e^{-\mu t} \end{aligned} \quad (3.23)$$

From Eq. (3.20) we obtain

$$\lambda \bar{a} \int_0^\infty a'(x) e^{-\mu x} e^{-\mu t} dx = \int_0^\infty n(x) e^{-\mu t} dx = \bar{n} e^{-\mu t} \quad (3.24)$$

where $\bar{n} = E[\text{no. found in system by a new arrival}]$. Using this expression for the first term on the right-hand side of Eq. (3.23) and taking Laplace transforms we obtain

$$s a^*(s) = \frac{s(\bar{n} + b + 1) + \mu}{s(s + \mu - \lambda \bar{a})} \quad (3.25)$$

Inverting, we get for $t \geq 0$ (observing that $a'(0) = \bar{n} + b + 1$),

$$a'(t) = \frac{1}{1-\rho} + \frac{(\bar{n} + b + 1)(1-\rho) - 1}{1-\rho} e^{-\mu(1-\rho)t} \quad (3.26)$$

where

$$\rho = \frac{\lambda \bar{a}}{\mu} \quad (3.27)$$

Here, \bar{n} and b are unknown quantities which need not be solved for directly. Instead we combine them in a new unknown $C = \bar{n} + b - \frac{\rho}{1-\rho}$ and we obtain

$$a'(t) = \frac{1}{1-\rho} + C e^{-\mu(1-\rho)t} \quad (3.28)$$

Integrating and using the initial condition that $a(0) = 0$ we get

$$a(t) = \frac{t}{1-\rho} + \frac{C}{\mu(1-\rho)} [1 - e^{-\mu(1-\rho)t}] \quad (3.29)$$

In the next section we will evaluate the constant C and calculate \bar{a} for a multilevel queueing system with RR at the last level where the service time distribution may be general up to this level, but must be exponential in this last (semi-infinite) level; i.e., $B(x)$ must have an exponential tail and we denote this system by M/GM/1. The same method can be used to complete the solution for a single level queue with bulk arrivals.

3.3.1b. M/GM/1 with Bulk Arrival. Returning to our discussion of the two level queueing system with breakpoints at a_{i-1} and a_i , we had Eq. (3.5)

$$T(a_{i-1} + \tau) = \frac{1}{1-\rho_{<a_{i-1}}} \left\{ W_{a_{i-1}} + a_{i-1} + a_2(\tau) \right\} \quad (3.5)$$

where $a_2(\tau)$ was the mean virtual time spent in the higher level queue. But in virtual time this is the solution for the bulk arrival case just studied. The study in the general case M/G/1 led to an integral equation, Eq. (3.21), for which no more explicit solution was obtained. However, in the case of an exponential distribution, we have the solution given in Eq. (3.29). Thus, we may permit RR at the first level (see Eq. (3.6)) in M/G/1 or at the last level in M/GM/1. In the latter case, we therefore consider the equivalent two-level system in which the breakpoints a_{i-1} and a_i are restricted to a_N and ∞ , respectively.

Thus, for the case $t = a_N + \tau$ we have from Eq. (3.29) that

$$a_2(\tau) = a(\tau) = \frac{\tau}{1-\rho} + d[1 - e^{-\mu(1-\rho)\tau}] \quad (3.30)$$

where $C = \mu(1-\rho)d$. Therefore, from Eq. (3.5),

$$T(a_N + \tau) = \frac{1}{1-\rho_{<a_N}} \left\{ W_{a_N} + a_N + \frac{\tau}{1-\rho} + d[1 - e^{-\mu(1-\rho)\tau}] \right\} \quad (3.31)$$

In addition to the constant d we also need to solve for \bar{a} , the mean size of the bulk arrivals to the RR queue, since this is contained in $\rho = \frac{\lambda \bar{a}}{\mu}$. This we do

for the general case a_{i-1}, a_i . \bar{a} is just the mean number of jobs that arrive during a lower level busy period and require more than a_{i-1} seconds of service. Therefore \bar{a} must satisfy the equation

$$\bar{a} = \lambda \bar{E}_{<a_{i-1}} \bar{a} + [1 - B(a_{i-1})] 1 \quad (3.32)$$

In this equation $\lambda \bar{E}_{<a_{i-1}}$ is the mean number of jobs that arrive during the service time of the first job in the busy period. Since each of these jobs in effect generates a busy period, there are an average of $\lambda \bar{E}_{<a_{i-1}}$ arrivals to the upper level queue due to these jobs. The second term is just the average number of times that the first job in the busy period will require more than a_{i-1} seconds of service.

Clearly then

$$\bar{a} = \frac{1 - B(a_{i-1})}{1 - \rho_{<a_{i-1}}} \quad (3.33)$$

Now we may complete the solution for $t = a_N + \tau$ by solving for C by conserving the mean work in the system. Since the single server works at a constant rate as long as there is any work in the system, the amount of work in the system at any time is independent of the service disciplines and system levels. It follows immediately that the mean work in the system is a constant (depending only on λ and the service time distribution). The mean work in the system is given by W_∞ (see Eq. (2.12)).

We also have from Eq. (3.20) that $n(t) = \lambda [1 - B(t)] T'(t)$ where $n(t)$ is the mean number of jobs in the system with t seconds of attained service. The mean remaining service requirement for a job which has already received t seconds of service is given by

$$\int_t^\infty \frac{x dB(x)}{1 - B(t)} - t \quad (3.34)$$

Therefore the mean work in the system is also given by

$$W_{\infty} = \int_0^{\infty} n(t) \left\{ \int_t^{\infty} \frac{x dB(x)}{1 - B(t)} - t \right\} dt \quad (3.35)$$

or

$$W_{\infty} = \int_0^{\infty} \lambda [1 - B(t)] T'(t) \left\{ \int_t^{\infty} \frac{x dB(x)}{1 - B(t)} - t \right\} dt \quad (3.36)$$

With no loss of generality, we may assume that the RR discipline is the lower level queue discipline. In this case we have from Eqs. (3.6), (3.31) and W_{∞} from Eq. (2.12) that

$$T(t) = \begin{cases} \frac{t}{1 - \rho_{< a_N}} & 0 \leq t \leq a_N \\ \frac{1}{1 - \rho_{< a_N}} \left\{ \frac{\lambda t^2 e^{-\lambda t}}{2(1 - \rho_{< a_N})} + a_N + \frac{t - a_N}{1 - \rho} \right. \\ \left. + d(1 - e^{-\mu(1-\rho)(t-a_N)}) \right\} & t > a_N \end{cases} \quad (3.37)$$

Using this expression for $T(t)$ in Eq. (3.36) we can solve for d . Since $B(x)$ is arbitrary in the range $0 \leq t \leq a_N$ (and exponential for $t > a_N$) the solution is not expressible in a compact form. When $B(x)$ is exponential over $0 < x < \infty$ the solution is simplified. In particular for all x

$$\int_t^{\infty} \frac{x dB(x)}{1 - B(t)} - t = \frac{1}{\mu} \quad (3.38)$$

Therefore

$$W_{\infty} = \int_0^{\infty} \lambda [1 - B(t)] T'(t) \frac{1}{\mu} dt = \frac{\lambda}{\mu} \int_0^{\infty} e^{-\mu t} T'(t) dt \quad (3.39)$$

Now using the Eq. (3.37) for $T(t)$ we have

$$W_{\infty} = \frac{\lambda}{\mu} \left\{ \frac{[1 - e^{-\mu a_N}]}{\mu(1 - \rho_{< a_N})} + e^{-\mu a_N} \frac{\lambda t^2 e^{-\lambda t}}{2(1 - \rho_{< a_N})^2} + \frac{e^{-\mu a_N}}{\mu(1 - \rho)(1 - \rho_{< a_N})} + \frac{C e^{-\mu a_N}}{\mu(2 - \rho)(1 - \rho_{< a_N})} \right\} \quad (3.40)$$

Setting $W_{\infty} = \frac{\lambda t^2 e^{-\lambda t}}{2(1 - \rho_{< a_N})} = \frac{\lambda \mu^2}{1 - \lambda/\mu}$ we essentially have a

solution for C . The solution is illustrated later in the examples section.

4. EXAMPLES

In this section we demonstrate through examples the nature of the results we have obtained. Recall that we have given explicit solutions for our general model in the case $M/G/1$ with processor sharing where the allowed scheduling disciplines within a given level may be either FCFS or FB. Moreover, for this general system we allow RR at level 1. That is, for the case $M/G/1$,

$$D_i = \begin{cases} RR, FCFS, FB & i = 1 \\ FCFS, FB & i = 2, 3, \dots, N+1 \end{cases} \quad (4.1)$$

Furthermore, in the case $M/G/1$ we permit

$$D_i = \begin{cases} RR, FCFS, FB & i = 1 \\ FCFS, FB & i = 2, 3, \dots, N \\ RR, FCFS, FB & i = N+1 \end{cases} \quad (4.2)$$

That is, we also permit RR at the highest level if $B(x)$ is of exponential form in the interval $a_N < x$.

We begin with three examples from the system $M/M/1$. As mentioned in Section 2, we have nine disciplines for the case $N = 1$. This comes about from Eq. (4.2) where we allow any one of three disciplines at level 1 and any one of three disciplines at level 2. As we have shown, the behavior of the average conditional response time in any particular level is independent of the discipline in all other levels. In Fig. 4.1 we show the behavior of each of the three disciplines for the system $N = 1$.

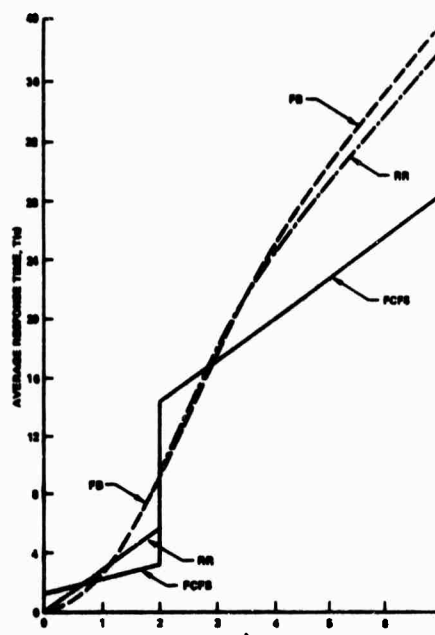


Figure 4.1. Response Time Possibilities for $N = 1$, $M/M/1$, $\mu = 1$, $\lambda = .75$, $a_1 = 2$

In this case we have assumed $\mu = 1$, $\lambda = 0.75$, and $a_1 = 2$.

Note that for the case $M/M/1$ we have from Eqs. (2.9), (2.10), and (3.33) the following:

$$\bar{E}_{< a_1} = \frac{1}{\mu} (1 - e^{-\mu a_1}) = 0.865 \quad (4.3)$$

$$\bar{t}_{< a_1}^2 = \frac{2}{\mu^2} [1 - e^{-\mu a_1} - \mu a_1 e^{-\mu a_1}] = 1.19 \quad (4.4)$$

$$\bar{a} = e^{-\mu a_1} / (1 - \lambda \bar{E}_{< a_1}) = 0.385 \quad (4.5)$$

Also, for the parameter values chosen, we have $C = 2.42$. From Eq. (3.1) we see that the response time for the FB system is completely independent of the values a_i and therefore the curve shown in Fig. 4.1 for this response time is applicable to all of our $M/M/1$ cases. Note the inflection point in this curve which results in a linear growth for response time as $t \rightarrow \infty$ (a phenomenon not observable from previously published figures). As can be seen from its defining equation, the response time for FCFS is linear regardless of the level; the RR system at level 1 is also linear, but as we see from this figure and from Eq. (3.37) the RR at level $N+1$ is non-linear. Thus one can determine the behavior of any of nine possible disciplines from Fig. 4.1. Adiri and Avi-Itzhak considered the case (FB, RR) [12].

Continuing with the case $M/M/1$, we show in Fig. 4.2 the case for $N = 3$ where $D_1 = RR$, $D_2 = FB$, $D_3 = FCFS$, and $D_4 = RR$. In this case we have chosen $a_1 = 1$ and $\mu = 1$, $\lambda = 0.75$. We also show in this figure the case FB over the entire range as a reference curve for comparison with this discipline. Note (in general for $M/M/1$) that the response time for any discipline in a given level must either coincide with FB curve or lie above it in the early part of the interval and below it in the latter part of the interval; this is true due to the conservation law [10].

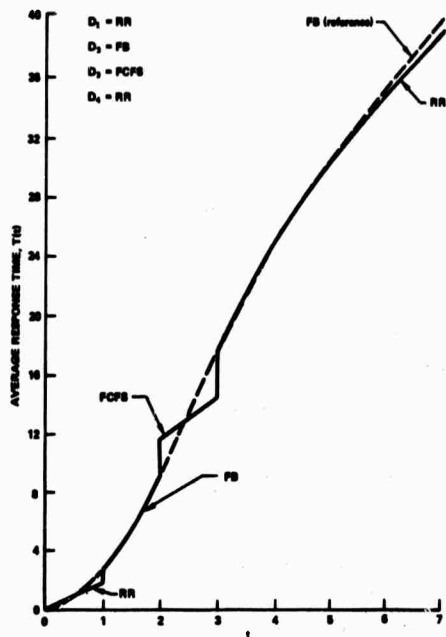


Figure 4.2. Response Time for an Example of $N = 3$, $M/M/1$, $\mu = 1$, $\lambda = 0.75$, $a_1 = 1$

The third example for $M/M/1$ is for the iterated structure $D_1 = FCFS$. Once again we have chosen $\mu = 1$, $\lambda = 0.75$, and $a_1 = 1$.

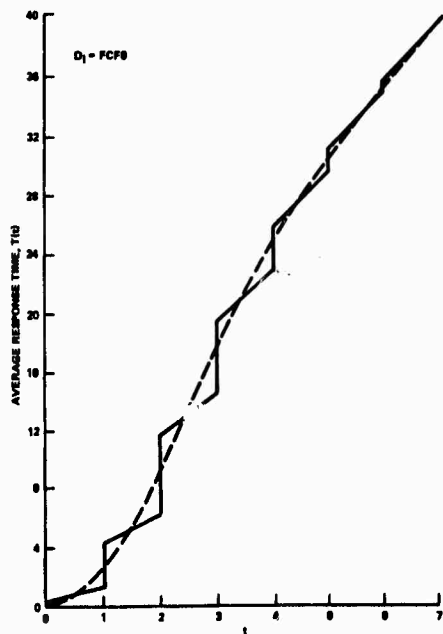


Figure 4.3. Response Time for the $M/M/1$ Iterated Structure, $\mu = 1$, $\lambda = 0.75$, $a_1 = 1$, $N = \infty$

Also shown in this figure is a dashed line corresponding to the FB system over the entire range. Clearly, one may select any sequence of FB and $FCFS$ with duplicates in adjacent intervals and the behavior for such systems can be found from Fig. 4.3. It is interesting to note in the general $M/G/1$ case with $D_1 = FCFS$ that we have a solution for the FB system with finite quantum $q_1 = a_1 - a_{1-1}$ where preemption within a quantum is permitted!

For our last example we choose the system $M/E_2/1$ with $N = 1$. In this system we have

$$\frac{dB(x)}{dx} = (2\mu)^2 x e^{-2\mu x} \quad x \geq 0 \quad (4.6)$$

as shown in Fig. 4.4. We note that the mean service time here is again given by $1/\mu$; the second moment of this distribution is $3/2\mu^2$. We calculate

$$E_{a_1} = \frac{1}{\mu} - \frac{1}{\mu} e^{-2\mu a_1} [1 + 2\mu a_1 + 2(\mu a_1)^2] \quad (4.7)$$

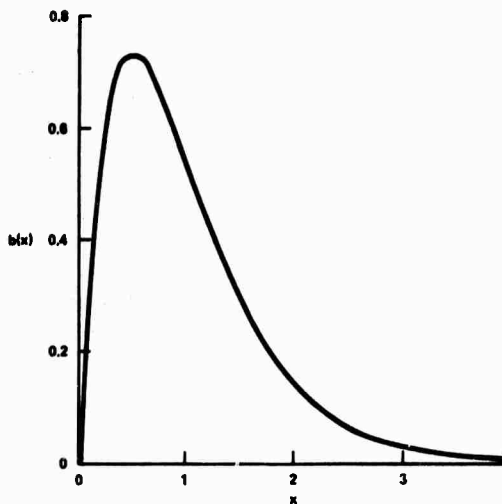


Figure 4.4. Service Time Density for $M/E_2/1$, $\mu = 1$

We choose the system $N = 1$ with $D_1 = RR$ and $D_2 = FCFS$. For the cases $a_1 = 1/2\mu, 1/\mu, 2/\mu, 4/\mu$ with $\mu = 1$ and $\lambda = 0.75$ we show in Fig. 4.5 the behavior of this system.

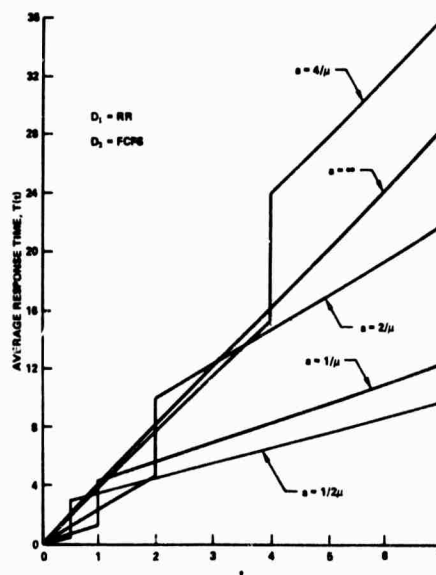


Figure 4.5. Response Time for $RR, FCFS$ in $M/E_2/1$ with $\mu = 1$, $\lambda = 0.75$ and $a = 1/2, 1, 2, 4, \infty$

This figure demonstrates again the kind of behavior obtainable from our results as one varies the appropriate system parameters; once again one may choose to discriminate in a variety of ways in favor of the short jobs and against the longer jobs.

5. CONCLUSION

Our purpose has been to generalize results in the modeling and analysis of time-shared systems. The class of systems considered was the processor-sharing systems in which various disciplines were permitted at different levels of attained service. The principle results for M/G/1 are the following: (1) The average conditional response time at level i is independent of the queueing discipline at all other levels; (2) the performance for the FC discipline at any level is given by Eq. (3.1); (3) the performance for the FCFS discipline is linear with t within any level and is given by Eq. (3.2); (4) the performance for the RR discipline at the first level is well-known [8] and is given by Eq. (3.6); (5) an integral equation for the average conditional response time for RR at any level (equivalent to bulk arrival RR) is given by Eq. (3.21) and remains unsolved in general. For M/GM/1 (exponential tail for $t > a_n$) we have the performance for RR at the last level as given by Eq. (3.37).

Examples are given which display the behavior of some of the possible system configurations. From these, we note the great flexibility available in these multilevel systems.

References

1. Kleinrock, L., "Analysis of A Time-Shared Processor," *Naval Research Logistics Quarterly*, Vol. II, No. 1, pp. 59-73, March 1964.
2. McKinney, J. M., "A Survey of Analytical Time-Sharing Models," *Computing Surveys*, Vol. 1, No. 2, June 1969, pp. 105-116.
3. Takacs, L., "A Single-Server Queue with Feedback," *Bell System Technical Journal*, March 1963, pp. 505-519.
4. Kleinrock, L., "Time-Shared Systems: A Theoretical Treatment," *JACM*, Vol. 14, No. 2, April 1967, pp. 242-261.
5. Kleinrock, L., and E. G. Coffman, "Feedback Queueing Models for Time-Shared Systems," *JACM*, Vol. 15, No. 4, October 1968, pp. 549-576.
6. Coffman, E.G., Jr., R.R. Muntz, and H. Trotter, "Waiting Time Distributions for Processor-Sharing Systems," *JACM*, Vol. 17, No. 1, Jan. 1970, pp. 123-130.
7. Kleinrock, L., and E. G. Coffman, "Distribution of Attained Service in Time-Shared Systems," *J. of Computers and Systems Science*, Vol. 3, October 1967, pp. 287-298.
8. Sakata, M., S. Noguchi, and J. Oizumi, "Analysis of a Processor-Shared Queueing Model for Time-Sharing Systems," *Proc., 2nd Hawaii International Conf. on System Sciences*, Jan. 1969, pp. 625-628.
9. Conway, R. W., W. L. Maxwell, and L. W. Miller, *Theory of Scheduling*, Addison-Wesley, 1967.
10. Kleinrock, L., "A Conservation Law for a Wide Class of Queueing Disciplines," *Naval Research Logistics Quarterly*, Vol. 12, No. 2, June 1965, pp. 181-192.
11. Schrage, L. E., "The Queue M/G/1 with Feedback to Lower Priority Queues," *Management Science*, Vol., 13, No. 7, 1967.
12. Adiri, I., and B. Avi-Itzhak, "Queueing Models for Time-Sharing Service Systems," *Operations Research, Statistics and Economics Mimeograph Series No. 27*, Technion, Israel.

APPENDIX D

BUFFER BEHAVIOR FOR BATCH POISSON ARRIVALS
AND SINGLE CONSTANT OUTPUT

by

Wesley W. Chu

BUFFER BEHAVIOR FOR BATCH POISSON ARRIVALS AND SINGLE CONSTANT OUTPUT

Wesley W. Chu
Member, IEEE

ABSTRACT

A queuing model with limited waiting room (buffer), batch (burst) Poisson arrivals, and a synchronous single server (transmission channel) with constant service time (constant transmission rate) is studied. Using average burst length and traffic intensity as parameters, the relationships among buffer size, overflow probabilities, and an approximation to the expected queuing delay of each burst due to buffering are obtained. These relationships are represented in graphs which provided a guide to the buffer design problem. A simple example is given which applies these results to the design of a buffer system in a time-sharing computer system. Although the problem discussed this paper arose in studies of design of statistical multiplexers, the queuing model developed is quite general, and may be useful for other industrial applications.

The author was with Bell Telephone Laboratories, Holmdel, N.J. He is now at the University of California, Los Angeles, California. This research is in part supported by Advanced Research Projects Agency of the Department of Defense (DAHC 15-69-0285) and the Office of Naval Research, N00014-69-A-02000-4027.

I. INTRODUCTION

Buffer design is one of the important considerations in communication systems design, such as multiplexing system, data compression system etc. The problem of interest is: For a given buffer input traffic characteristics, buffer output transmission rate, what are 1) the relationship between the overflow probability (the average fraction of the total number of arriving data rejected by the buffer) and buffer size at various traffic intensities, and 2) the expected queuing delay due to buffering. Birdsall¹ and later Dor² have analyzed the buffer behavior of Poisson input arrivals, and constant output. Chu³ have further studied the buffer behavior of this model with multiple outputs. In many data communication systems, however, the input traffic are in bursts (strings of characters) rather than in single characters. For example, in computer communication systems, the computer output traffic are in bursts. From measurements of some operating computer systems, Fuchs and Jackson⁴ have found that the bursts arrivals can be approximated as Poisson distributed and the burst length can be approximated as geometrically distributed. The buffer behavior with bursty input traffic are very different from that of the previous analyzed models. Therefore, in this paper, a finite waiting room queuing model is used to study the buffer behavior with bursty traffic inputs. The results obtained from this study are portrayed in graphs which provided a guide to the buffer design problem.

II. ANALYSIS OF BUFFER BEHAVIOR

Let us define the time to transmit a character from the multiplex line as a unit service interval and denoted as μ . For a line with a transmission rate R characters per second, then $\mu = 1/R$ seconds per character. Next, we assume the burst length, X , is geometrically distributed with mean, $\bar{l} = 1/\theta$, and the number of bursts arrived during a unit service interval Y , is Poisson distributed with a rate of λ bursts per service time. The density functions of X and Y are as follows:

$$f_X(\ell) = \theta(1 - \theta)^{\ell-1} \quad \ell = 1, 2, \dots \quad (1)$$

$$f_Y(n) = \exp(-\lambda) \lambda^n / n! \quad n = 0, 1, 2, \dots \quad (2)$$

The total number of characters that arrived during the time to transmit a character on the multiplexed line is a random sum and equals

$$S_Y = \sum_{i=0}^Y X_i \quad (3)$$

where X_i , a random variable distributed as (1), is the number of characters contained in the i^{th} arriving burst, and Y , a random variable distributed as (2), is the total number of bursts arriving during the unit service interval. For simplicity in notation, we let $S \equiv S_Y$.

The characteristic function of S , $\phi_S(u)$, can be expressed in terms of the characteristic function of X , $\phi_X(u)$, and λ .

$$\phi_S(u) = \exp[-\lambda + \lambda \phi_X(u)] \quad (4)$$

Since the burst lengths are geometrically distributed, the characteristic function of X is

$$\phi_X(u) = \theta \exp(iu) / (1 - (1 - \theta) \exp(iu)) \quad (5)$$

where $i = \sqrt{-1}$. Substituting (5) into (4), then

$$\phi_S(u) = \exp[-\lambda + \lambda \theta \exp(iu) / (1 - (1 - \theta) \exp(iu))] \quad (6)$$

From (6), it can be shown that the probability density function of j characters arriving during a unit service interval, $\Pr(s = j) = \pi_j$, is a compound Poisson distribution as shown in the following:

$$\pi_j = \begin{cases} \sum_{k=1}^j \binom{j-1}{k-1} (\lambda \theta)^k (1-\theta)^{j-k} \exp(-\lambda) / k! & j = 1, 2, \dots \\ \exp(-\lambda) & j = 0 \end{cases} \quad (7)$$

The expected value of S is $E[S] = E[X] \cdot E[Y] = \lambda/\theta$, and the variance of S is

$$\text{var}[S] = \lambda(2 - \theta)/\theta^2 \quad (8)$$

The time required to compute π_j from (7) is dependent on the size of j . For larger j (e.g., $j > 1000$), the computation time could be very large and prohibitive. A convenient and less time consuming way to compute π_j is from $\phi_S(u)$ by using the Fast Fourier Transform inversion method⁵ as follows:

$$\pi_j = \sum_{r=1}^M \phi_S(r) \exp[-2\pi i r j / M] \quad j = 0, 1, 2, \dots, M-1 \quad (9)$$

where

$$r = 2\pi u / M$$

M = total number of input points to represent $\phi_S(r)$
= total number of output values of π_j .

In order to accurately determine $\phi_S(r)$, it is computed with double precision on the IBM 360/65. Further, we would like to use as many points as possible to represent $\phi_S(r)$; that is, we would like to make M as large as possible. Because of the word length limitation of the computer, double precision provides 15-digit accuracy. Therefore, when $\pi_j < 10^{-15}$, it is set equal to zero. M is selected such that $\pi_j > M < 10^{-15}$. The M 's are different for different values of λ and $\bar{\ell}$.

To establish the set of state equations for a buffer size of N characters, we assume that the system has reached its equilibrium. Let p_n be

the probability that there are exactly n characters in the buffer at the end of a service interval.[†] Without loss of generality, we can express the probability of the number of characters in the buffer at the end of the unit service interval in terms of the probability of the number present at the end of the last interval, multiplied by the probability of a given number of characters arriving during the service interval. As this can occur in different combinations, we add the probabilities.

$$p_n = \pi_0 p_{n+1} + \sum_{i=1}^n \pi_{n-i+1} p_i + \pi_n p_0$$

or

$$p_{n+1} = \frac{1}{\pi_0} \left[p_n - \sum_{i=1}^n \pi_{n-i+1} p_i - \pi_n p_0 \right]$$

$$n = 0, 1, 2, \dots, N-1 \quad (10)$$

$$\sum_{i=0}^N p_i = 1 \quad (11)$$

and

$$p_{i \geq N+1} = 0$$

Equation (10) states that after a service interval, there are n characters in the buffer if: 1) there were $n+1$ characters in the buffer at the beginning of the service interval, and no character arrived during the service interval;

[†]In queuing theory terminology, the system modeled is one in which there is a gate between the waiting room and the server, which is opened (and instantaneously closed) at fixed intervals. Hence, p_n should be understood as the number of characters (customers) in the buffer at the end of a service interval and immediately before the gate is opened.

or 2) i characters in the buffer at the beginning of the service interval, and $n - i + 1$ characters arrived during the service interval, $i = 1, 2, \dots, n$; or 3) the buffer is empty and n characters arrived during the service interval.

Since the buffer has a finite size of N , $p_i \geq N+1 = 0$. As a result, when a character arrives and the buffer is full, an overflow will result. Consequently, the average character departure-rate from the buffer (carried load), α , is less than the average character arrival-rate to the buffer (offered load), $\beta = \lambda/\theta$. The carried load can be computed from the probability that the buffer is idle,

$$\alpha = 1 - p_0 \quad (12)$$

The overflow probability of the buffer, the average fractions of the total number of arriving characters rejected by the buffer, is then equal to

$$P_{of} = \frac{\text{offered load} - \text{carried load}}{\text{offered load}} = 1 - \alpha/\beta \quad (13)$$

Traffic intensity, ρ , measures the degree of congestion and indicates the impact of a traffic stream upon the service streams. It is defined as the ratio of offered load and unit service interval. Thus,

$$\rho = \beta/\mu = \lambda/(\theta\mu) = \lambda \bar{l}/\mu \quad (14)$$

Channel (server) utilization, u , measures the fraction of time that the lines are busy. It can be expressed as

$$u = (1 - P_{of}) \beta/\mu = \alpha/\mu \leq \rho \quad (15)$$

Since physically it is impossible for the transmission line to be more than 100 per cent busy, the utilization is limited to a numerical value less than unity. In the no-loss case, (unlimited buffer size), $P_{of} = 0$, then $u = \rho$.

The set of state Equations (10) is an imbedded Markov Chain. The state probabilities can be solved iteratively and expressed in terms of p_0 .

The value of p_0 can be determined from (11). Thus we can find all the state probabilities. The overflow probabilities for various burst lengths can then be computed from (13). Numerical results are presented in Figures 1 through 6. Figure 7 provides the relationship (at $P_{of} = 10^{-6}$) between burst lengths and buffer sizes for selected traffic intensities.

In the above analysis, we have treated each character as a unit. However, in computing the expected burst delay, D , due to buffering, we should treat each burst as a unit. The service time is now the time required to transmit the entire burst. For a line with constant transmission rate, the service time distribution is the same as the burst length distribution except by a constant transmission rate factor. Hence, the service time is also geometrically distributed. When the overflow probability is very small, a good approximation for the expected delay can be computed from a queuing model with finite waiting room, Poisson arrivals and single server with geometric service time. The expected burst delay⁶ can be expressed in terms of the expected number of bursts in the buffer, L , and the effective burst arrival rate, $\lambda_{eff} = \lambda(1 - P_{of})$. When overflow probability is very small, for example, $P_{of} = 10^{-6}$, then $D = L/\lambda_{eff} = L/\lambda$ which implies that D can be approximated by the expected burst delay of the infinite waiting room, M/G/1, model.⁷ Hence

$$D = \frac{\lambda E(X^2)}{2(1-\rho)} = \frac{\lambda(2-\theta)}{2(\theta-\lambda)\theta} \text{ character-service times} \quad (16)$$

where $E(X^2)$ = second moment of burst length, X . The delays are computed from (16) for selected traffic intensities and burst lengths. Their relationships are portrayed in Figure 8.

III. DISCUSSION OF RESULTS

An interesting property of the state probabilities is that for large n (for example, $n \geq 10$), the state probabilities p_n and p_{n+1} are related by

a constant; that is, $p_{n+1}/p_n = K$, where K is a function of traffic intensity and expected burst length as shown in Table I.

TABLE I
VALUES OF K 's AT VARIOUS ρ 's AND \bar{l} 's

$\bar{l} \backslash \rho$	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1	0.127	0.198	0.285	0.388	0.509	0.650	0.813
2	0.619	0.664	0.712	0.763	0.817	0.874	0.935
4	0.818	0.842	0.867	0.892	0.918	0.944	0.972
6	0.880	0.897	0.913	0.930	0.947	0.964	0.982
8	0.911	0.932	0.936	0.948	0.961	0.974	0.987
10	0.935	0.939	0.949	0.959	0.969	0.979	0.990
20	0.964	0.970	0.975	0.979	0.985	0.990	0.995
40	0.982	0.985	0.987	0.989	0.992	0.994	0.997
60	0.988	0.989	0.992	0.993	0.994	0.996	0.998

The overflow probability depends upon the buffer size, the traffic intensity, and expected burst length. For a given average buffer length, the overflow probability increases as the traffic intensity increases (see Figures 1-6). For a given traffic intensity, as the average burst length increases, the buffer size has to increase in order to achieve the desired overflow probability as shown in Figure 7.

In the above analysis, we have assumed that the burst length is geometrically distributed which takes values from zero to infinity. In practice, however, the maximum burst length is limited to a finite number of characters. Because of the long tail effect of the geometric distribution, the result we obtain here will be more conservative than that of the truncated geometric distribution.

When the average burst length equals unity, then the result reduces to the case of Poisson arrivals and constant service time as had been analyzed by Birdsall,¹ and Dor.² For a given traffic intensity, the required buffer size for average burst lengths \bar{l} ($\bar{l} > 1$), $N_{\bar{l}}$ to achieve the same degree of overflow probability is much greater than that for unity burst length, N_1 . In general, $N_{\bar{l}} > \bar{l} \times N_1$. As \bar{l} increases, the difference between $N_{\bar{l}}$ and $\bar{l} \times N_1$ increases. For example, for $\rho = .8$,

$\bar{l} = 1$, the required buffer size to achieve $P_{of} = 10^{-6}$, is $N_1 = 28$ characters. When $\bar{l} = 4$, then from Figure 7, $N_4 = 212 > 4 \times 28 = 112$ characters. In the manner, if $\bar{l} = 20$, $N_{20} = 1200 > 20 \times 28 = 560$ characters. This is due to the fact that the variance of S is proportional to \bar{l} as shown in (8). Figure 8 portrays the relationship between expected burst queuing delay and traffic intensity for selected expected burst lengths. For a given expected burst length, the expected queuing delay increases as traffic intensity increases; for a given traffic intensity, the expected queuing delay increases with burst length. These are important factors that affect the delay.

In the above model, the data output from the buffer is assumed to be transmitted on a synchronous transmission system, i.e., the data are taking out synchronously from the buffer for transmission at each discrete clock time. The data arriving at the buffer during the period between clock times have to wait to begin transmission at the beginning of the next clock time, even the transmission facility is idle at the time of arrival. In queuing theory terminology, the above system implies there is a gate between the server and waiting room which is opened at fixed intervals. The result derived from the above system can also be used as a conservative estimate (upper bound) for the case in which there is no gate between the server and waiting room, i.e., the line is permitted to transmit the characters arrived during the service interval. The estimate yield better approximation for the heavy than light traffic intensity case. Because under heavy traffic case, the line is usually busy and the characters that arrive during the service interval have to wait and cannot be serviced during the service interval. The maximum over design of the buffer system with a single transmission line that permits to transmit characters arrived during service interval is one character.

IV. EXAMPLE

Consider the design of a time-sharing computer system with many remote terminals which employs the Asynchronous Time Division Multiplexing Technique⁸ to transmit data from the central processor to the

user terminals (Figure 9). A typical computer-to-user data stream is shown in Figure 10. Measuring the traffic statistics from several operating systems⁴ revealed that the burst interarrival time from central processor to each user can be approximated as exponentially distributed with a mean of 2.84 seconds; that is, the bursts can be approximated to be generated from a Poisson process with a rate of $\lambda = 0.35$ bursts/sec. Further, the burst length can also be approximated as geometrically distributed with a mean of $\bar{\ell} = 20$ characters. All the terminals are assumed to be independent and to have identical traffic characteristics. A reasonable and conservative estimate is that 20 per cent of the transmitted information is to be used for addressing and framing. Under these conditions, suppose a full duplex transmission line that transmits 480 char/sec is used to provide communications from the central processor to the 46 terminals, then the traffic intensity $\rho = 1.2 \cdot 46 \cdot \lambda \cdot \bar{\ell} / \mu \approx 0.8$. To achieve an overflow probability of 10^{-8} , from Figure 7, we find that the required buffer size is 1,650 characters. From Figure 8, the expected queuing delay for each burst is 80 character-service times, or $80/480 = 0.167$ seconds.

Suppose now we changed our transmission rate from 480 to 960 char/sec; then the traffic intensity $\rho \approx 0.4$. The corresponding required buffer size in order to achieve an overflow probability of 10^{-8} is 580 characters, and the delay is 13 character-service times or 14 milliseconds. Thus, these results provide insight regarding the trade off between transmission costs and storage costs as well as the relationship between expected burst queuing delays with transmission rates.

ACKNOWLEDGMENT

The author wishes to thank S. P. Bader of Bell Telephone Laboratories for his programming assistance.

BLANK PAGE

REFERENCES

1. T.G. Birdsall, et. al., "Analysis of Asynchronous Time Multiplexing of Speech Sources," IRE Trans. on Communications Systems, pp. 390-397, December 1962.
2. N.M. Dor, "Guide to the Length of Buffer Storage Required for Random (Poisson) Input and Constant Output Rates," IEEE Trans. on Electronic Computer, pp. 683-684, October 1967.
3. W.W. Chu, "Buffer Behavior for Poisson Arrivals and Multiple Synchronous Constant Outputs," IEEE Transaction on Computers, June 1970.
4. E. Fuchs and P.E. Jackson, "Estimates of Distributions of Random Variables for Certain Computer Communication Traffic Models," Proceedings of ACM Symposium on Problems in the Optimization of Data Communications Systems, pp. 201-227, Pine Mountain, Georgia, October 13-16, 1969.
5. W.M. Gentleman and G. Sande, "Fast Fourier Transforms - For Fun and Profit," Proc. AFIPS 1966 Fall Joint Computer Conference, Vol. 29, Spartan Books, New York, pp. 563-578.
6. N. U. Prabhu, Queues and Inventories, John Wiley & Sons, Inc., p. 42, 1965.
7. J.D.C. Little, "A Proof of the Queuing Formula, $L = \lambda W$," Opns. Res. 9, pp. 383-387, 1961.
8. W.W. Chu, "A Study of the Technique of Asynchronous Time-Division Multiplexing for Time-Sharing Computer Communications," Proceedings of Second Hawaii International Conference on System Sciences, pp. 607-610, Honolulu, Hawaii, January 22-24, 1969.

KEY PHRASES

Queuing Theory

Buffer Design

Time-Sharing Systems

Computer Communications

Asynchronous Time Division Multiplexing

Overflow Probabilities

Queuing Delay

Statistical Multiplexor

Concentrating Multiplexor

FIGURE CAPTIONS

- Figure 1 Buffer Length Versus Overflow Probabilities, $\ell = 2$ Characters
- Figure 2 Buffer Length Versus Overflow Probabilities, $\ell = 4$ Characters
- Figure 3 Buffer Length Versus Overflow Probabilities, $\ell = 10$ Characters
- Figure 4 Buffer Length Versus Overflow Probabilities, $\ell = 20$ Characters
- Figure 5 Buffer Length Versus Overflow Probabilities, $\ell = 40$ Characters
- Figure 6 Buffer Length Versus Overflow Probabilities, $\ell = 60$ Characters
- Figure 7 Buffer Length Versus Average Burst Length, $P_{of} = 10^{-6}$
- Figure 8 Traffic Intensity Versus Expected Burst Queuing Delay
- Figure 9 Asynchronous Time Division Multiplexing System for
Time-Sharing Computer Communications
- Figure 10 Asynchronous Time Division Multiplexing Data Stream

BLANK PAGE

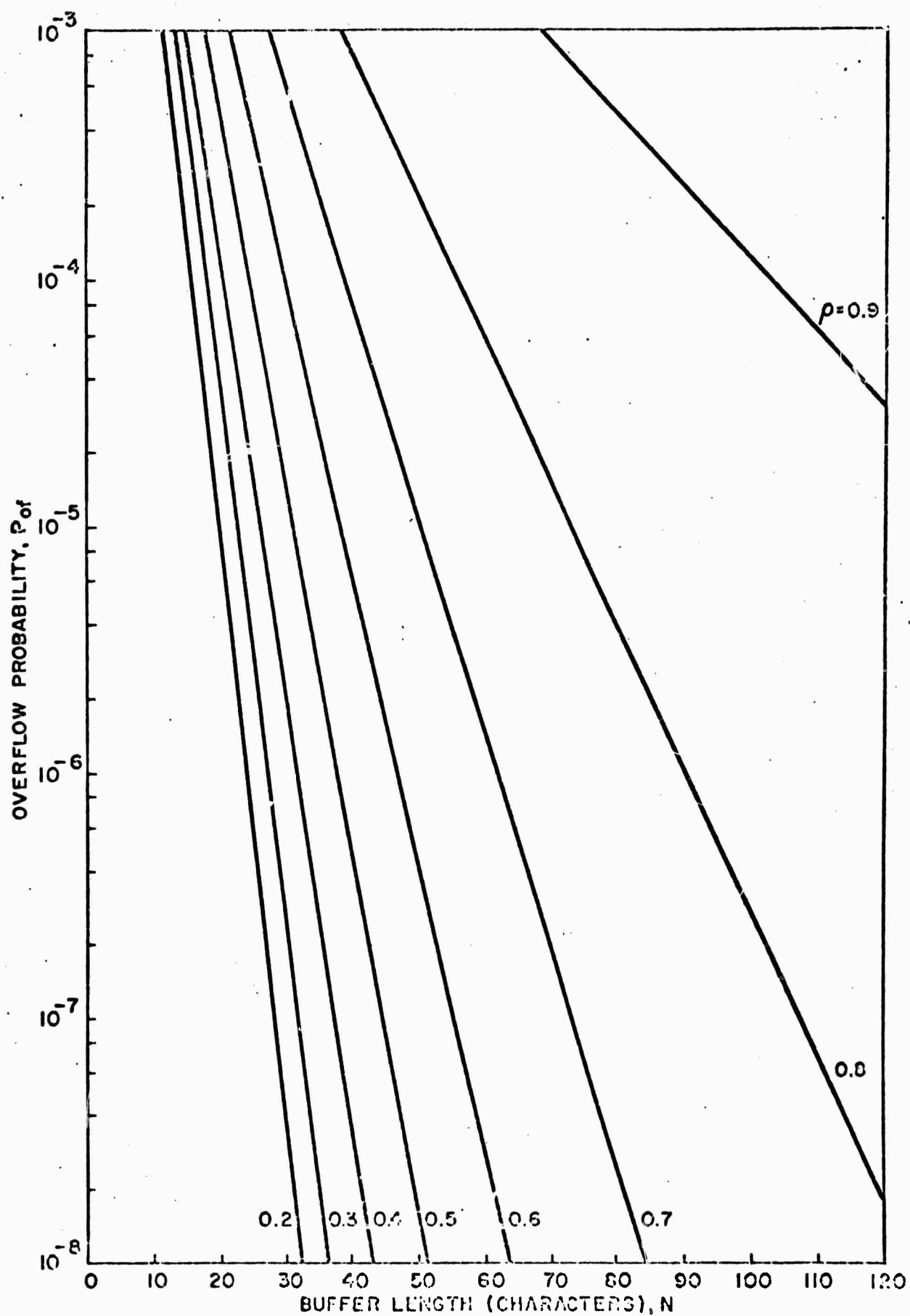


FIG. 1 BUFFER LENGTH VS OVERFLOW PROBABILITIES, $l = 2$ CHARACTERS

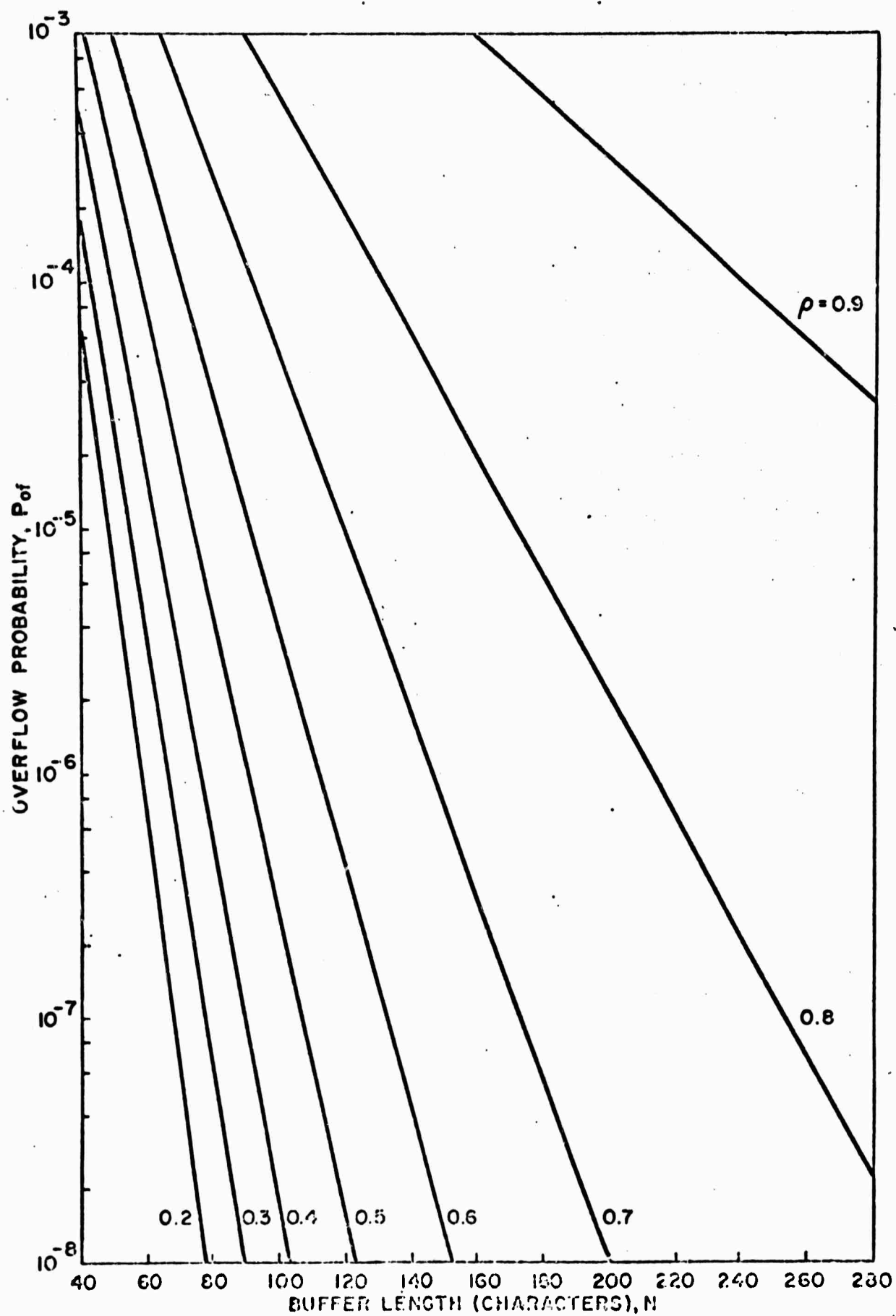


FIG. 2 BUFFER LENGTH VS OVERFLOW PROBABILITIES, $L = 4$ CHARACTERS

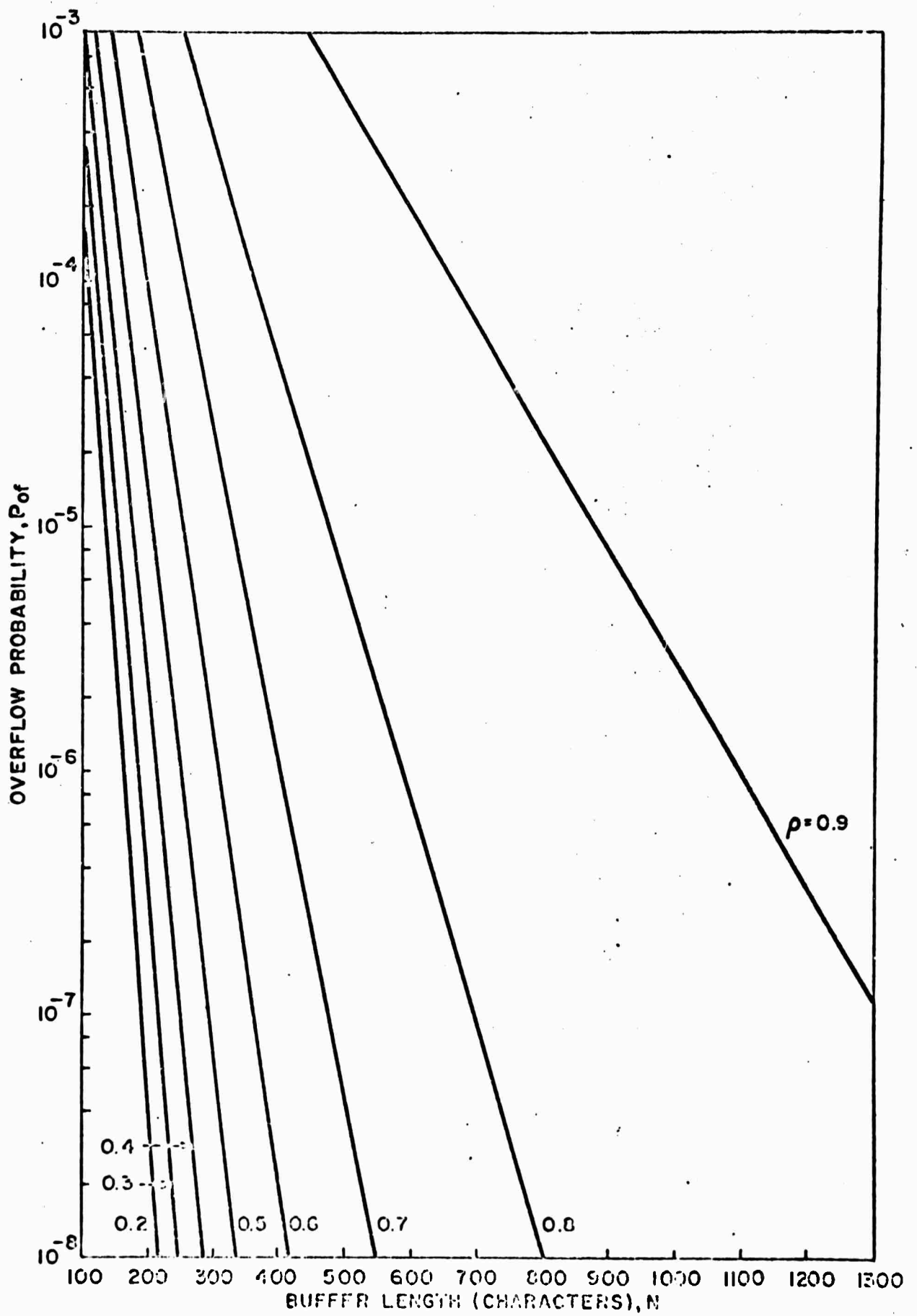


FIG. 3 BUFFER LENGTH VS OVERFLOW PROBABILITIES, $l = 10$ CHARACTERS

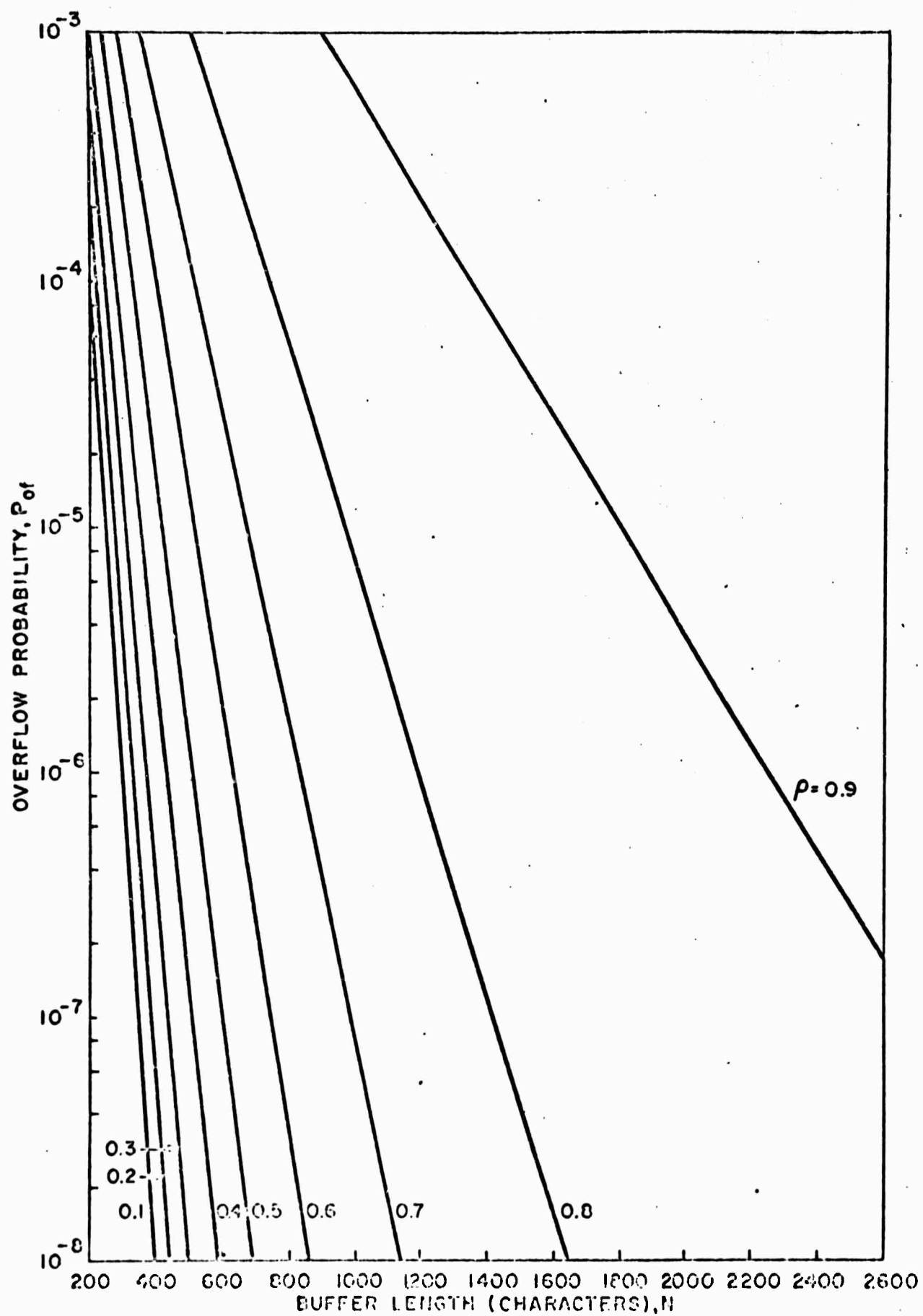


FIG.4 BUFFER LENGTH VS OVERFLOW PROBABILITIES, $l = 20$ CHARACTERS

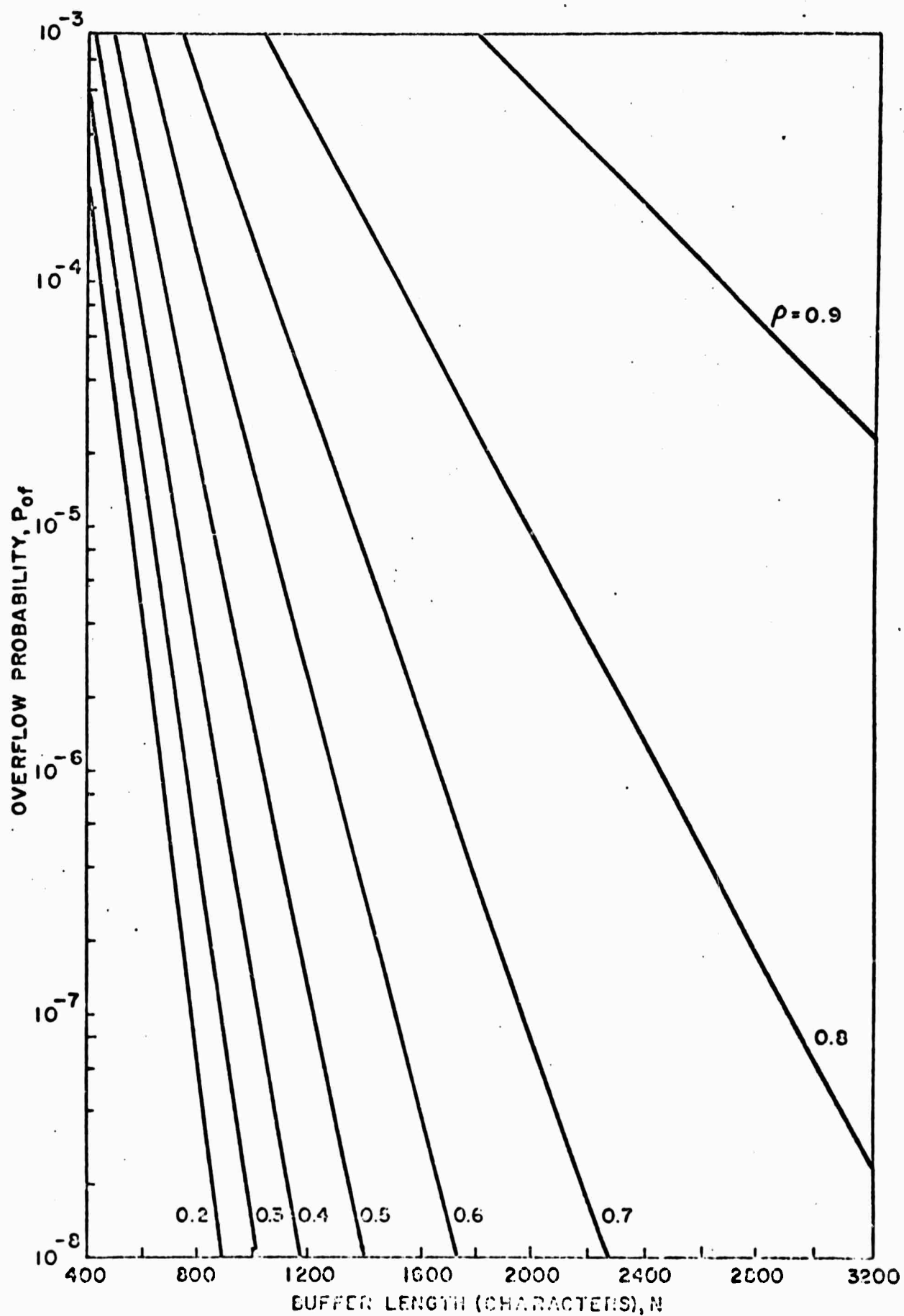


FIG.5 BUFFER LENGTH VS OVERFLOW PROBABILITIES, $L = 40$ CHARACTERS

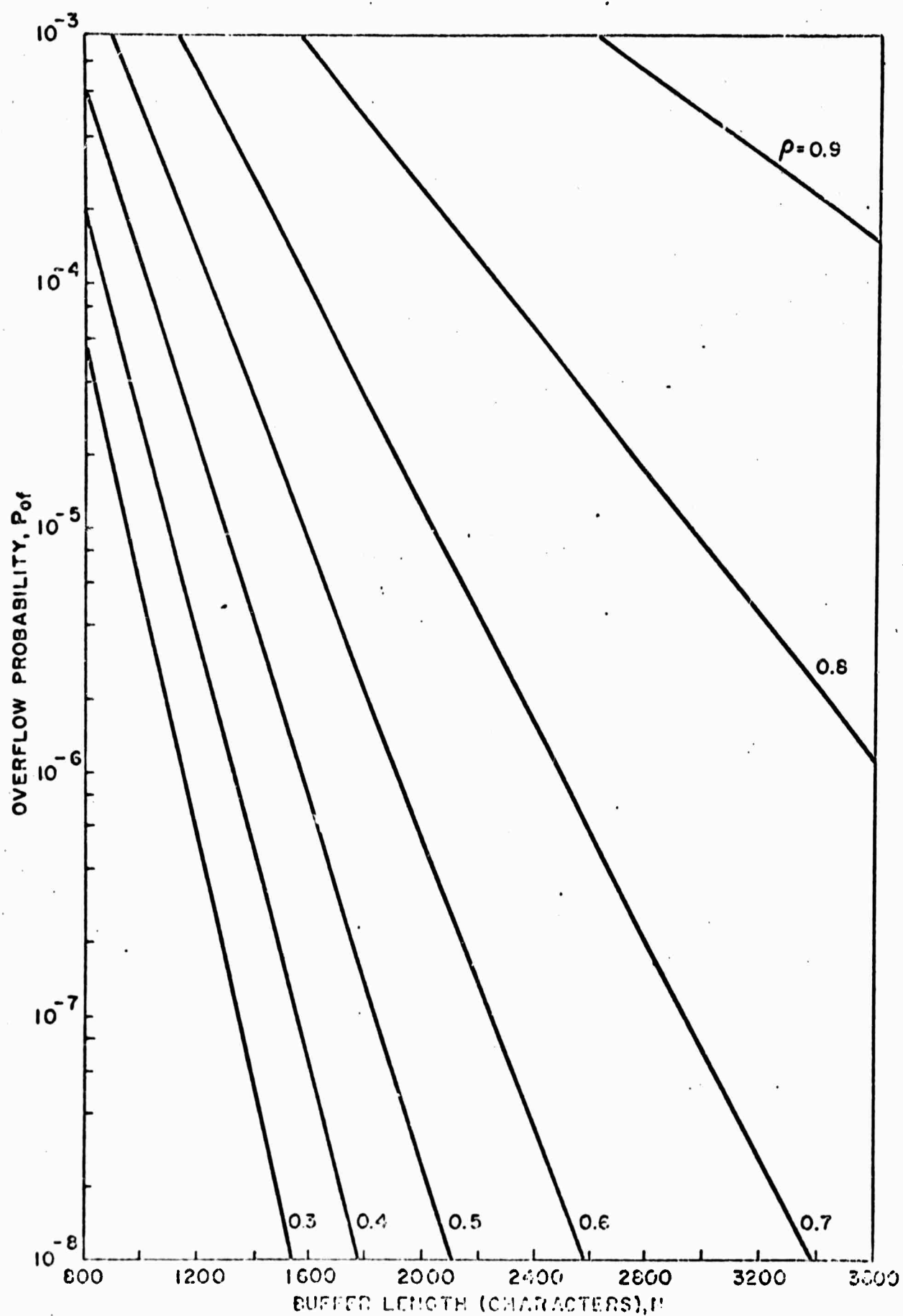


FIG. 6 BUFFER LENGTH VS OVERFLOW PROBABILITIES, $L = 60$ CHARACTERS

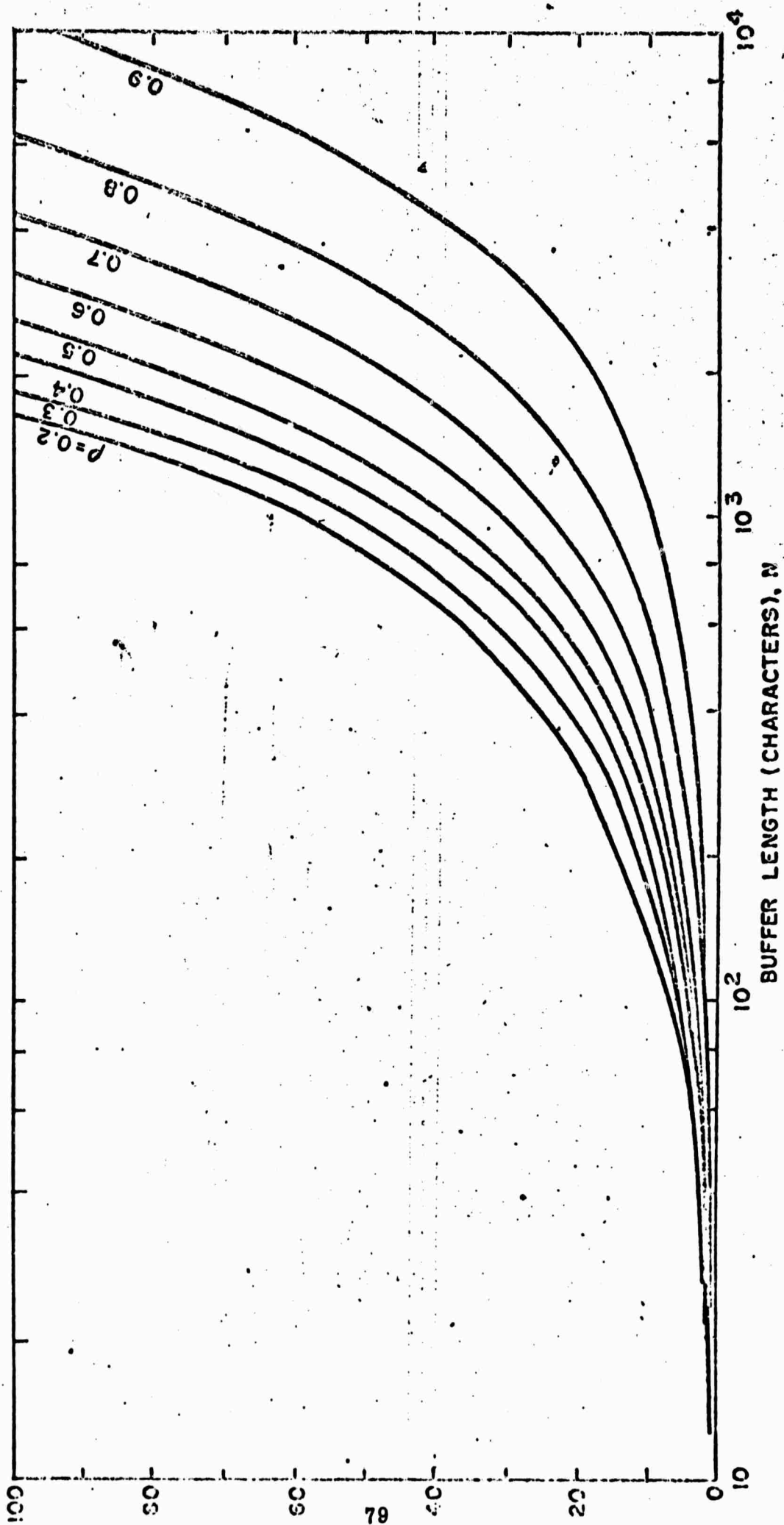


FIG.7 BUFFER LENGTH VS AVERAGE BURST LENGTH, $\rho_{of} = 10^{-6}$

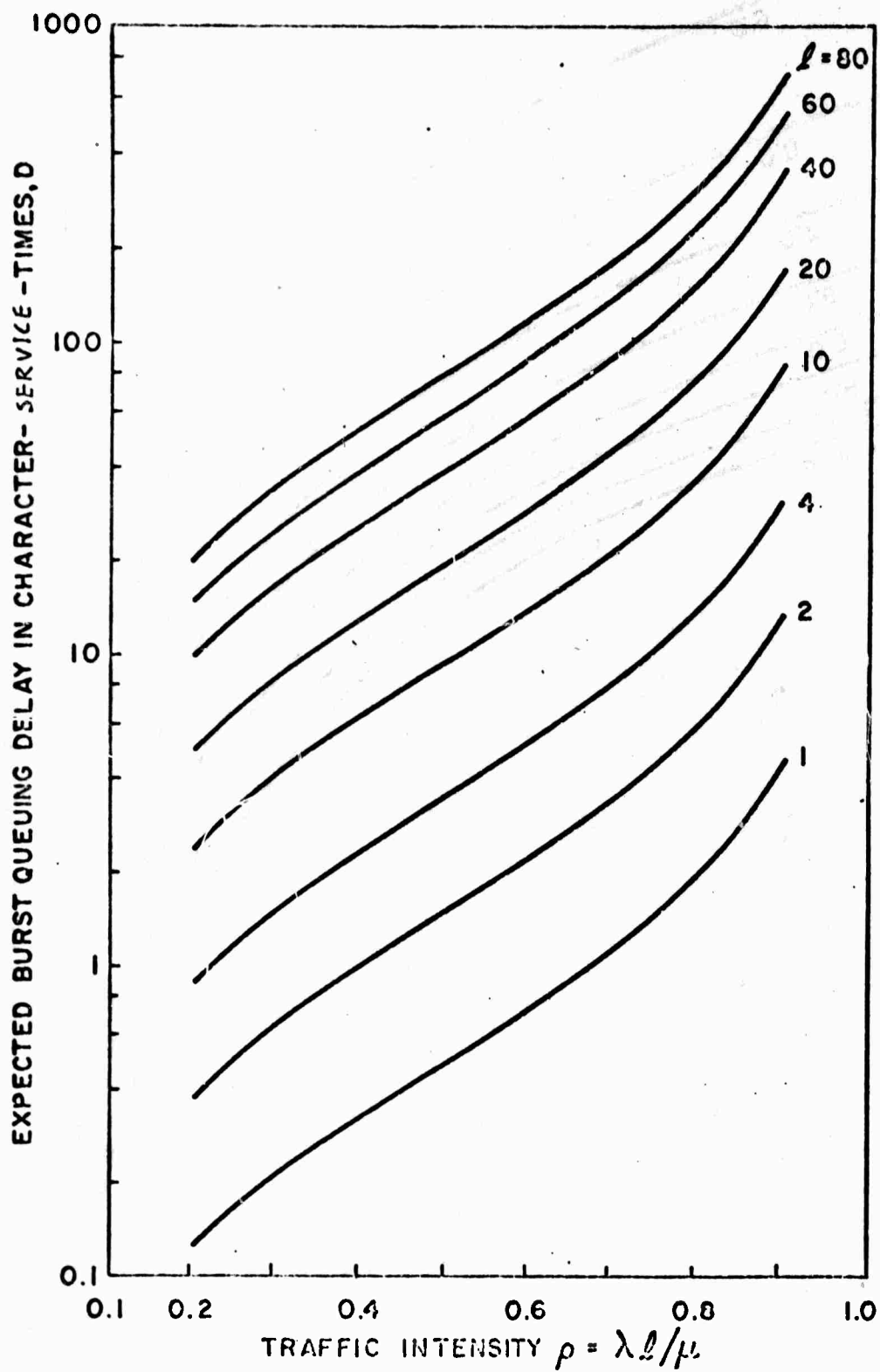


FIG. 2 TRAFFIC INTENSITY VS EXPECTED BURST QUEUING DELAY

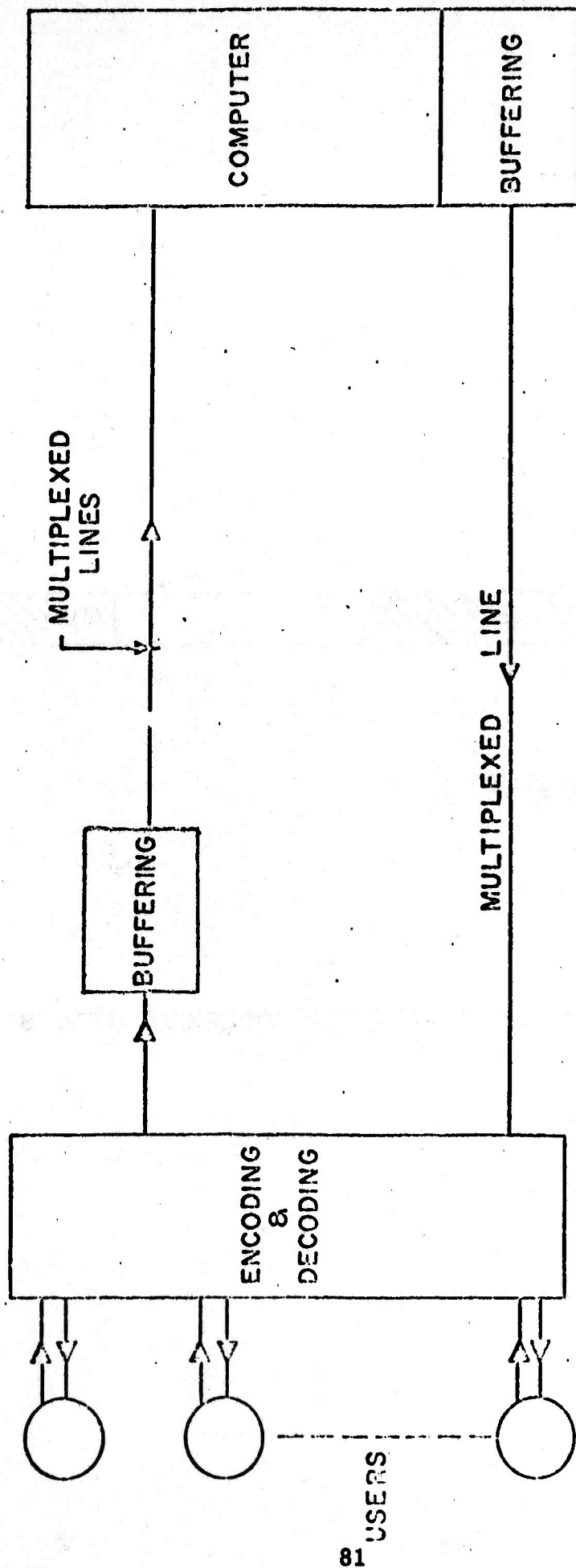


FIG. 9 ASYNCHRONOUS TIME DIVISION MULTIPLEXING SYSTEM FOR TIME-SHARING
COMPUTER COMMUNICATIONS




ADS	ADDRESS
E	END OF MESSAGE
	MESSAGE

FIG. 10 ASYNCHRONOUS TIME DIVISION MULTIPLEXING DATA STREAM

APPENDIX E

SELECTION OF OPTIMAL TRANSMISSION RATE
FOR STATISTICAL MULTIPLEXERS

by

Wesley W. Chu

BLANK PAGE

SELECTION OF OPTIMAL TRANSMISSION RATE FOR STATISTICAL MULTIPLEXORS*

Wesley W. Chu

Computer Science Department
School of Engineering and Applied Science
University of California
Los Angeles, California

Summary

In the design of statistical multiplexors for computer communication networks or time sharing systems, one of the important considerations is the design trade-off between transmission rate and buffer size. For a given traffic arrival rate and required system performance (buffer overflow probability and buffer queuing delay), the higher the transmission rate of the channel, the lower the required buffer size but higher communication cost. Assuming the relationships among overflow probability, queuing delay, and buffer size are known, a model is developed in this paper to determine the optimal transmission rate for a statistical multiplexing system such that the operating cost (defined as the sum of communication costs and storage costs) is minimum yet also satisfies the required system performance.

Introduction

In order to reduce communications cost in computer communications systems, the Asynchronous Time Division Multiplexing technique has been proposed.^{1,2} In designing such a statistical multiplexing system, one of the important cost trade-off considerations is the selection of optimal transmission rate for the communications channel.³ Traffic intensity, ρ , is commonly used to measure the congestion of the line and can be computed as the ratio of the input traffic load, β , to the transmission rate of the channel, R . For a given β (in same unit as R), a higher transmission rate channel yields a lower traffic intensity, and thus requires a smaller buffer size. On the other hand, a lower transmission rate channel yields a higher traffic intensity and requires a larger buffer size.^{2,3} Thus, there is a trade-off between communications cost and buffer storage cost in the design of statistical multiplexors. Assuming the relationships among overflow probability, expected queuing delay due to buffering, traffic intensity, and buffer size are known, and we are given the input traffic load and communication distance d , the problem is to operate the multiplexing

*This research was supported by the Advanced Research Projects Agency of the Department of Defense (DAHC 15-69-C-0285).

system at an optimal traffic intensity. This problem is equivalent to selecting the transmission rate of the communication channel such that the operating cost (defined as the sum of transmission cost and storage cost) is minimized subject to achieving required multiplexing performance, e.g., that overflow probability and average queuing delay due to buffering be less than certain values.

In this paper, an analytical model is developed to analyze this problem.

Analysis

The operating cost of a statistical multiplexing system, C , can be separated into two parts — communication cost and storage cost. For a given distance, the communications cost depends upon the transmission rate of the channel. Hence, for a given input traffic load and distance, the transmission cost is related to the traffic intensity of the channel. The higher the traffic intensity, i.e., the lower the transmission rate, the lower the communications cost. Thus, the communications cost per month, $C_t(\rho)$ is a monotonic decreasing function of ρ as shown in Figure 1. Further, $C_t(\rho)$ can be easily generated from β and the channel costs of the set of available transmission rates. Since there is a practical upper limit on the available transmission rates, ρ can never reach zero. We shall denote the minimum ρ as ρ_{\min} . To represent $C_t(\rho)$ as a mathematical expression, the least squares polynomial curve fitting technique⁵ may be used. i.e.,

$$C_t(\rho) = a_0 + a_1\rho + a_2\rho^2 + \dots + a_{i-1}\rho^{i-1} + a_n\rho^n$$

$$\text{for } \rho_{\min} \leq \rho < 1 \quad (1)$$

where a_i = constant coefficients, $i = 1, 2, \dots, n$. Since only a limited number of transmission rates exist or planned, the set of points $\{(\rho_j, C_t(\rho_j)) \mid j = 1, 2, \dots\}$ can be determined and used to solve for the a_i 's in (1) for each distance d .

To achieve a certain probability of overflow, P_{of} , at a certain traffic intensity ρ , the required buffer size, $N(\rho, P_{of})$ is dependent upon the message characteristics and the channel transmission rate. The quantitative relationships

of $N(\rho, P_{of})$ with ρ and P_{of} can be obtained from an analysis of a queuing model with a finite waiting room. For example, when the messages can be approximated as Poisson arrivals and the message length is geometrically distributed, the buffer behavior has been analyzed in Reference 2. The relationship between buffer size with traffic intensity at a given probability of overflow can be obtained. Few typical of such relationships are portrayed in Figure 2. We note that the required buffer size increases with traffic intensity and is a monotonic increasing function of ρ . Further, the required buffer size increases when allowable buffer overflow probability decreases.

The storage cost per month is then equal to the monthly cost* of a unit of storage, C_b , multiplied the required buffer size, $N(\rho, P_{of})$, i.e., $C_s(\rho) = C_b \cdot N(\rho, P_{of})$. To represent $N(\rho, P_{of})$ as a mathematical expression at a given overflow probability, we shall again use the least squares polynomial fitting technique. Thus,

$$C_s(\rho) = C_b \cdot [b_0 + b_1\rho + b_2\rho^2 + \dots + b_m\rho^m] \quad (2)$$

for $\rho_{\min} \leq \rho < 1$

where b_i = constant coefficients, $i=1, 2, \dots, m$.

For a given input traffic load, required buffer overflow probability, and transmission distance, the operating cost is the algebraic sum of (1) and (2). Thus,

$$C(\rho) = C_s(\rho) + C_t(\rho) \quad \text{for } \rho_{\min} \leq \rho < 1 \quad (3)$$

We wish to minimize $C(\rho)$ with respect to ρ and subject to the delay requirements; that is,

$$C = \min_{\rho} C(\rho) = \min_{\rho} [C_s(\rho) + C_t(\rho)]$$

subject to $D(\rho) \leq D_a$ for $\rho_{\min} \leq \rho < 1$ (4)

where

$D(\rho)$ = expected queuing delay of each message due to buffering

D_a = maximum allowable expected queuing delay of each message

Note that the overflow probability requirement is satisfied automatically as (2) is represented from the corresponding overflow probability.

Since in almost all the queuing models, $D(\rho)$ is a monotonic increasing function of ρ ,

*The monthly cost of a unit of storage can be computed from the life of the system.

$D(\rho) \leq D_a$ is equivalent to $\rho \leq \rho_{\max}$ where ρ_{\max} is the maximum allowable traffic intensity such that the expected queuing delay is still less than or equal to the maximum allowable delay. ρ_{\max} can be obtained either from the buffer delay characteristic curves^{1,2} or from the analytical expression of expected delay function. For most multiplexing system, the buffer is designed to allow very small overflow probability - less than 10^{-4} . Thus, using the analytical expression of expected queuing delay derived from infinite waiting room to determine ρ_{\max} yields a good approximation. Thus (4) can be further reduced to

$$\min_{\rho} C(\rho) = \min_{\rho} [\alpha_0 + \alpha_1\rho + \dots + \alpha_l\rho^l] \quad (5)$$

for $\rho_{\min} \leq \rho \leq \rho_{\max}$

where $l = \max\{m, n\}$

$$\alpha_i = \begin{cases} a_i + b_i \cdot C_b & \text{for } 0 \leq i \leq k, k = \min\{m, n\} \\ a_i & i > k \text{ and } n > m \\ b_i \cdot C_b & i > k \text{ and } n < m \end{cases}$$

Let us now study the property of (5). Geometrically we know that $C_s(\rho)$ is a monotonic decreasing function of ρ . Thus the sum of these two functions, $C(\rho)$ is a convex function of ρ as shown in Figure 3. The least squares fitting of the functions $C_s(\rho)$ and $C_t(\rho)$ might introduce noise in the function, however, if enough terms are used to represent these functions, $C(\rho)$ can be approximated as a convex function of ρ . This assures a unique optimal traffic intensity, ρ^0 , exists that yields minimum operating cost.

There are many methods that can be used to numerically solve (5) and to obtain ρ^0 , such as convex nonlinear programming,⁶ geometric programming⁷ (if all the α_i 's are positive), or finding the set of ρ 's that satisfy $[dC(\rho)/d\rho] = 0$ and $\rho_{\min} \leq \rho \leq \rho_{\max}$. ρ^0 is the one that minimizes the cost function (5).

The optimal channel transmission rate for the statistical multiplexor, R^0 , can be computed from ρ^0 and input traffic load β ; that is,

$$R^0 = \beta/\rho^0 \quad \text{bits/sec} \quad (6)$$

Discussion of Model

From (3), we noticed that for the case in which the transmission distance between computers and/or terminals is very short, then the communication cost, compared to the storage cost, is relatively low. As a result, the storage cost becomes the dominant factor that

influences the selection of the optimal transmission rate. Under this case, the statistical multiplexor is said to be storage bound. On the other hand, when the transmission distance is very long, then the transmission cost becomes the dominant factor that affects the selection of optimal channel transmission rate. The multiplexor is then said to be channel bound. Under certain transmission distance, (e.g. Figure 3) both transmission cost and storage cost carry equal weight in the selection of the optimal transmission rate, the multiplexor is then said to be operated under balanced condition.

Although the model introduced in this paper is mainly for a statistical multiplexor that has a single transmission line output, it can be generalized to that of the multiple transmission lines case. $C_t(\rho)$ in (3) should now be viewed as the total transmission cost of the multiple transmission lines as a function of traffic intensity. $C_s(\rho)$ is the storage cost of the buffer (with multiple outputs) to achieve a desired level of probability of overflow. For a given system configuration; i.e., the number of output transmission lines and their transmission rate, based on the above model, the optimal transmission rate and the corresponding operating cost can be determined. We are now not only treat the design trade-off between storage cost and transmission cost, but also should consider the trade-off between various system configurations such as whether we should use more slower transmission rate lines or fewer high transmission rate lines. The optimal system configuration is the one that yields minimum operating cost among the set of possible system configurations.

Conclusion

An analytical model has developed to treat the cost trade-off between buffer storage cost and channel transmission cost. The choice of optimal channel transmission rate depends on many parameters, such as the input traffic load, transmission distance, required buffer overflow probability, maximum allowable expected queuing delay, cost of unit of storage, transmission cost per miles per month, and availability of channel transmission rates. These parameters closely interact with each other. The analytical

formulation of the model can also be solved geometrically, as shown in Figure 3. By properly selecting the channel transmission rate, the statistical multiplexor will operate at optimal traffic intensity and so yields the minimum operating cost yet satisfying the system performance requirements. The model should provide a useful guide line in selecting the optimal channel transmission rate for the Asynchronous Time Division Multiplexing systems.

References

1. W. W. Chu, "A Study of Asynchronous Time Division Multiplexing Technique for Computer Communications," Proceedings of the Second International Conference on System Science, pp. 607-610, Jan. 22-24, 1969, Honolulu, Hawaii.
2. W. W. Chu, "A Study of Asynchronous Time Division Multiplexing Technique for Time Sharing Computer Systems," Proceedings of FJCC, Vol. 35, pp. 669-678, 1969.
3. W. W. Chu, "Design Considerations of Statistical Multiplexors," Proceedings of ACM Symposium on Problems in the Optimization of Data Communication Systems, Pine Mountain, Georgia, pp. 36-60, Oct. 13-16, 1969.
4. R. G. Gould, Comments on "Generalized Cost Expressions for Private-Line Communications Channels," IEEE Transactions on Communications Technology, Sept. 1965, pp. 374-377.
5. Hamming, Numerical Methods for Scientists and Engineers, McGraw-Hill Book Company.
6. G. Hadley, Nonlinear and Dynamic Programming, Addison-Wesley Publishing Co., Inc., 1964.
7. R. J. Duffin, E. L. Peterson, C. M. Zener, Geometric Programming, John Wiley & Sons, Inc., 1967.

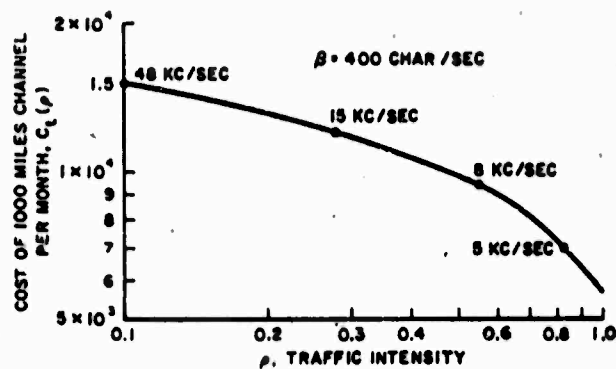


Figure 1 $C_t(\rho)$ vs ρ

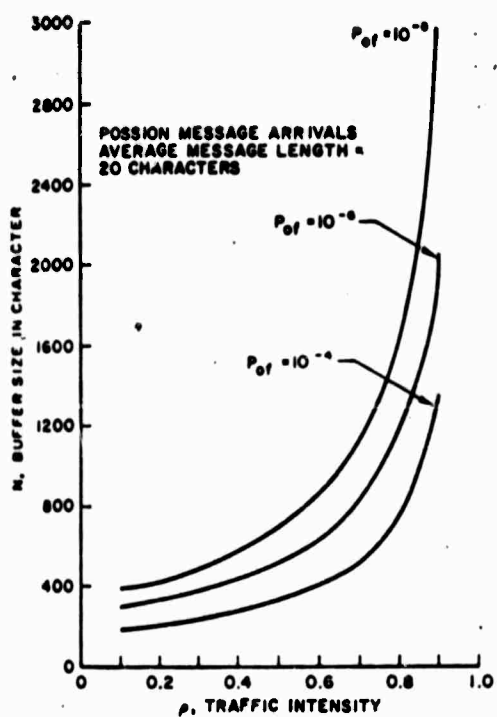


Figure 2 $N(\rho, P_{of})$ vs ρ

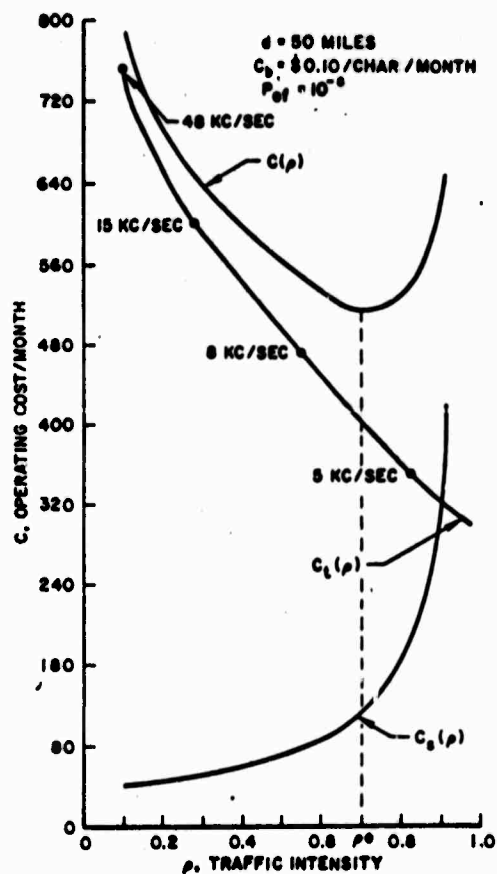


Figure 3 $C(\rho), C_t(\rho), C_s(\rho)$ vs ρ

APPENDIX F

COMPUTER NETWORK STUDIES

by

Gary Fultz

BLANK PAGE

APPENDIX F

COMPUTER NETWORK STUDIES

In the study of the performance of networks of computers such as the ARPA Network, the basic performance of the total system will be governed, on the lowest level, by the message handling capability of the store-and-forward communication network interconnecting the computers. In order to analyze the message handling capability of the network, message routing strategies within the network must be examined. Before routing is examined, one must first be able to model and analyze the network performance either mathematically or via simulation.

From the above requirements, we have determined six areas which should be investigated to determine the communication network message handling capability. These areas are:

1. Communication network performance measures
2. Store-and-forward communication network modelling
3. Message routing techniques
4. Message flow control and acknowledgment procedures
5. Computer network simulation
6. Network topological design

Of these six items, we have actively pursued investigation of the first five items and are now beginning investigation of the sixth. The following discussion and presentation of results will, for the most part, assume a fixed network topology with 50 KBPS communication links, with IMPS as nodes, and with HOSTS as the sources and sinks of messages.

As Kleinrock pointed out, one of the basic goals in the design of communication networks is to

$$\text{minimize } T = \sum_{i=1}^N \frac{\lambda_i}{\gamma} T_i \quad (1)$$

$$\left\{ \begin{array}{l} \text{capacity assignment} \\ \text{routing procedure} \\ \text{priority discipline} \\ \text{topology} \end{array} \right\}$$

subject to a cost constraint. Here T is the average message delay, λ_i is the average number of messages per second arriving at channel i , T_i is the delay suffered at channel i , and γ is the total arrival rate of messages entering the network from external sources.

Before the minimization of T can be carried out, we must first determine the validity of Eq. (1) in predicting the average message delay for store-and-forward communication networks and then understand how the minimization parameters effect T .

I. Store-and-Forward Communication Network Performance Measures

We have formulated communication network performance measures which can be related to either mathematical models or to simulation experiments. The measures are:

1. Average message delay for the network
2. Average message delay for a particular source HOST-destination HOST pair

3. Bounds on the P_r (average message delay $> T_0$).
4. Node storage requirements.
5. Packet blockage due to finite node storage.
6. How the routing doctrine, queue structures, message priorities, node storage size, network connectivity, and message blocking effect message throughput in the network.

In order to calculate those parameters which effect the performance measures, we must be able to model the network in order to make it amenable to mathematical analysis.

II. Store-and-Forward Communication Network Modelling

Listed below are some of the models we have been considering and the analytical results we have obtained.

A. Average Message Delay Model

The basic problem is to calculate the average message delay as given by Eq. (1). We have assumed a priority system where acknowledgments have the highest priority, singlepacket messages the next lowest priority, and long message, the lowest priority. The traffic matrix entries are given at a design point. We will then consider scaling all entries in the traffic matrix by RE (the effective rate). RE = 1 corresponds to the design point. We will also assume that the mix of short messages and long messages is given by M and 1 - M, respectively.

1. Short Message Model (maximum of one packet each)

The equation for the calculation of the average message delay is given by

$$T = \sum_{i=1}^N \frac{R_i}{C} \left[\frac{\mu_P}{C} + W_i(R_i, R'_i) + TL_i + T_{cpu} \right] + T_{cpu} \quad (2)$$

where

$$W_i(R_i, R'_i) = \frac{\rho_{A_i} \frac{\mu_A}{C} + \rho_{P_i} \frac{\mu_P}{C} (1 + C_P^2)}{(1 - \rho_{A_i}) (1 - \rho_{A_i} - \rho_{P_i})}$$

and

$$\rho_{P_i} = \frac{R_i}{C} \quad \rho_{A_i} = \frac{\mu_A}{\mu_P} \frac{R'_i}{C}$$

R_i is the data rate in line i and R'_i is the data rate flowing in the opposite direction of the fully duplexed line i

$$R_T = \sum_{i=1}^N R_i$$

C = Channel capacity of all lines in bits per sec.

μ_P = E [Packet Length] in bits

C_P = Packet coefficient of variation

μ_A = Acknowledgment length in bits

TL_i = Propagation time of line i

And T_{cpu} = The average processing time of a packet by the cpu routine in each node.

Figures 1 through 4 show the theoretical and experimental results for T versus RE for four different network configurations. These results show that Eq. (2) is a reasonably accurate prediction of the average message delay.

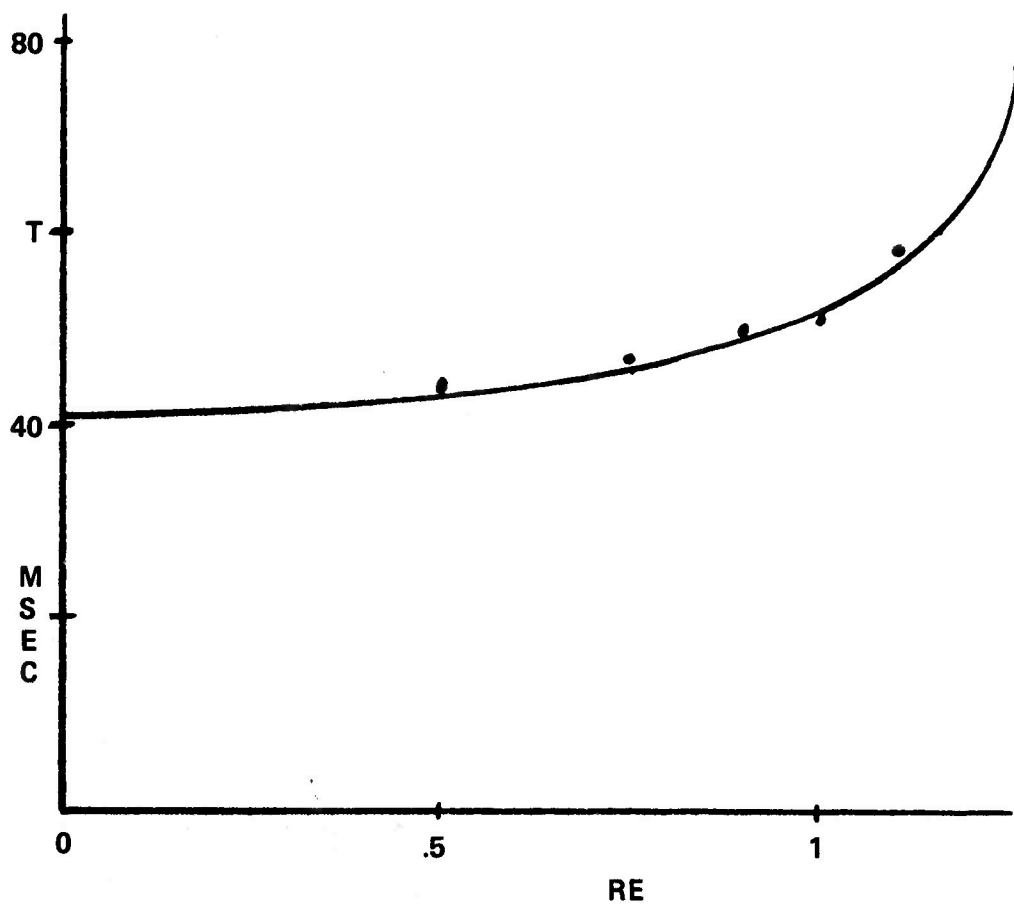
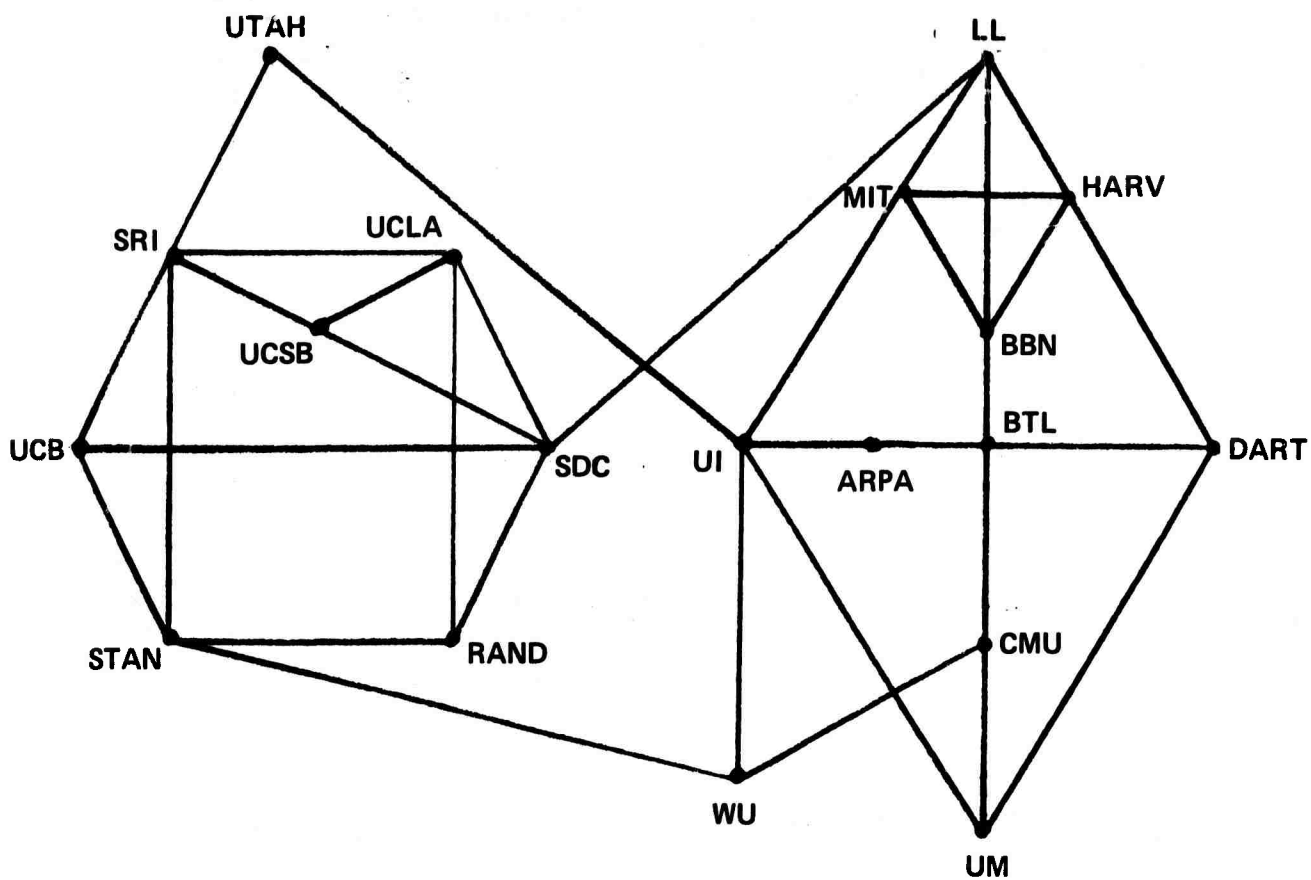


Figure 1. 19 Node Net

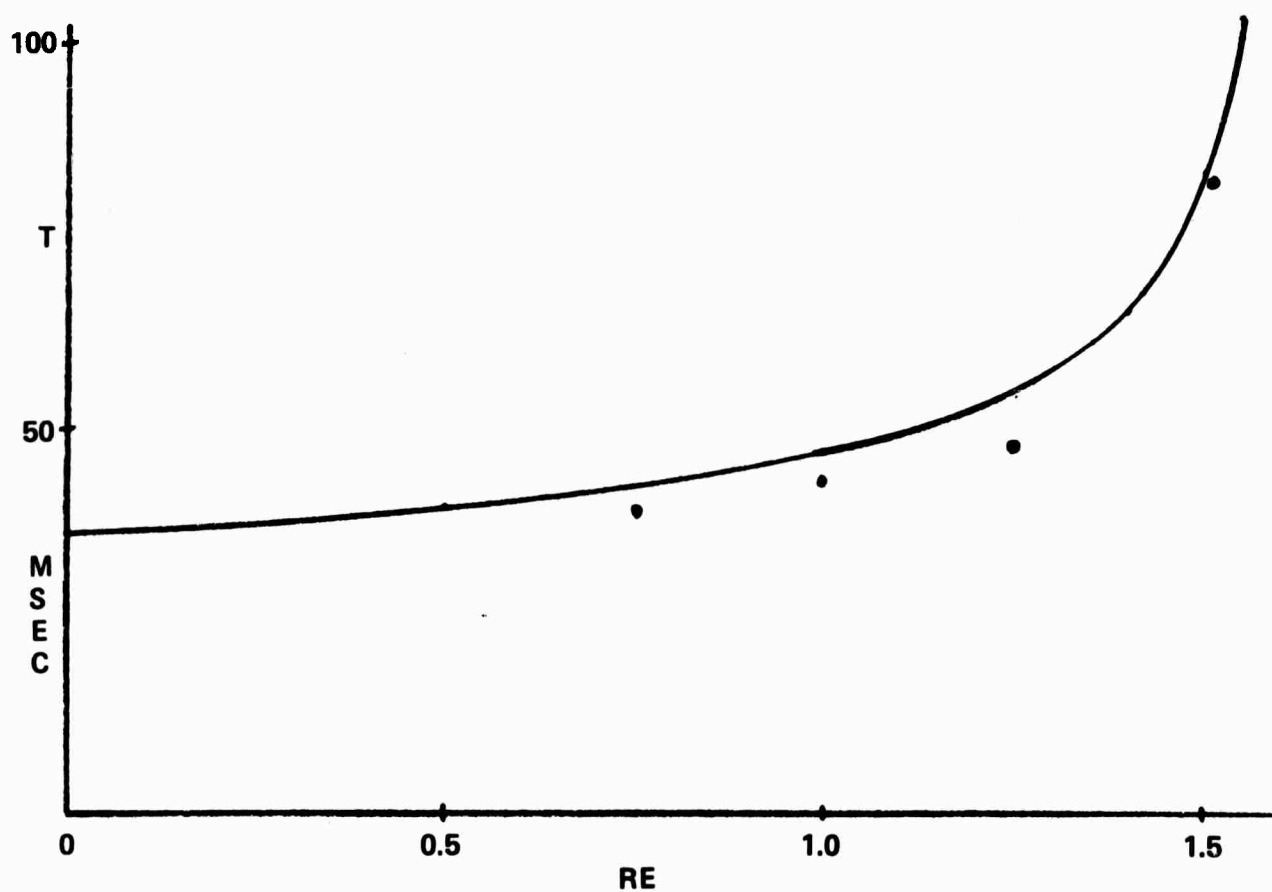
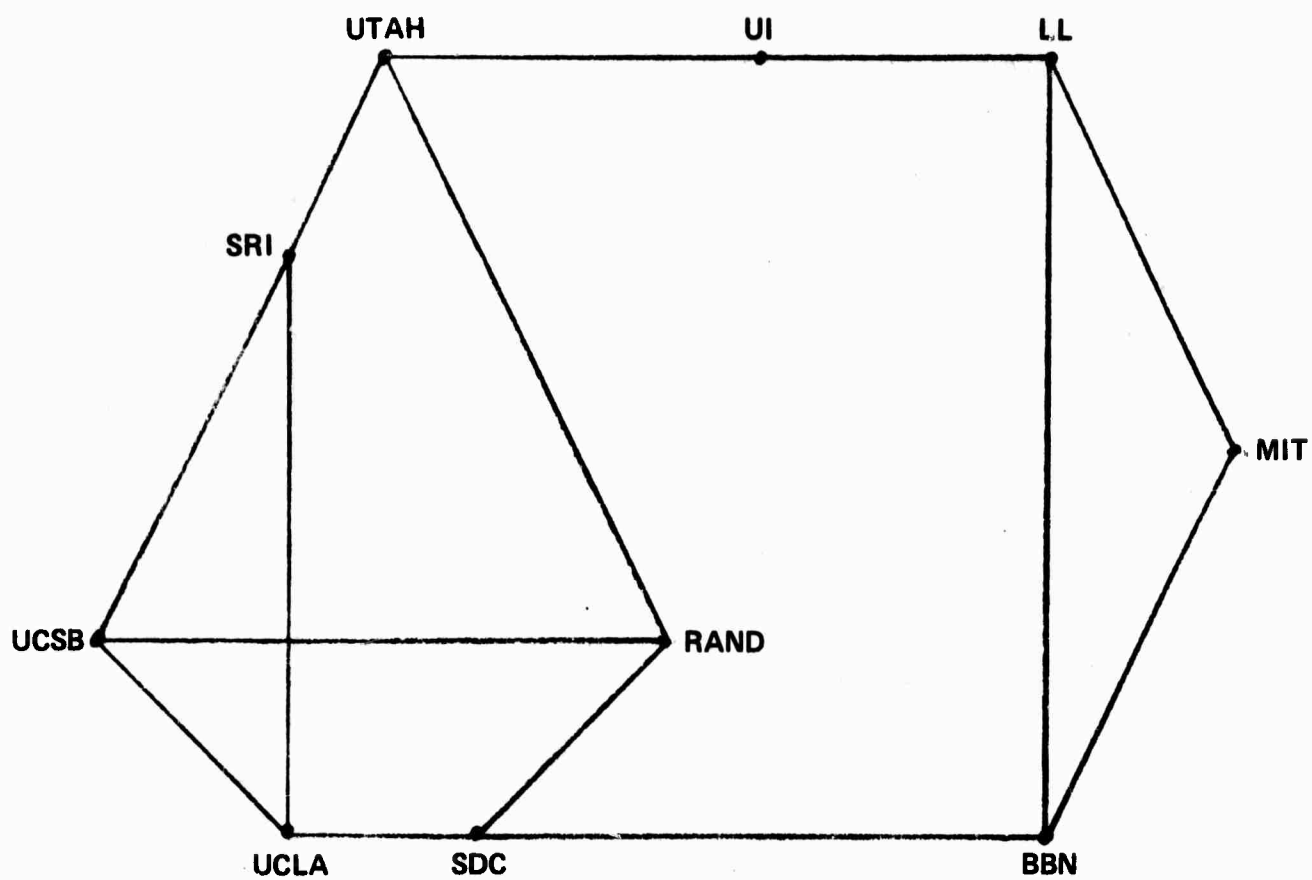


Figure 2. 10 Node Net--Configuration A

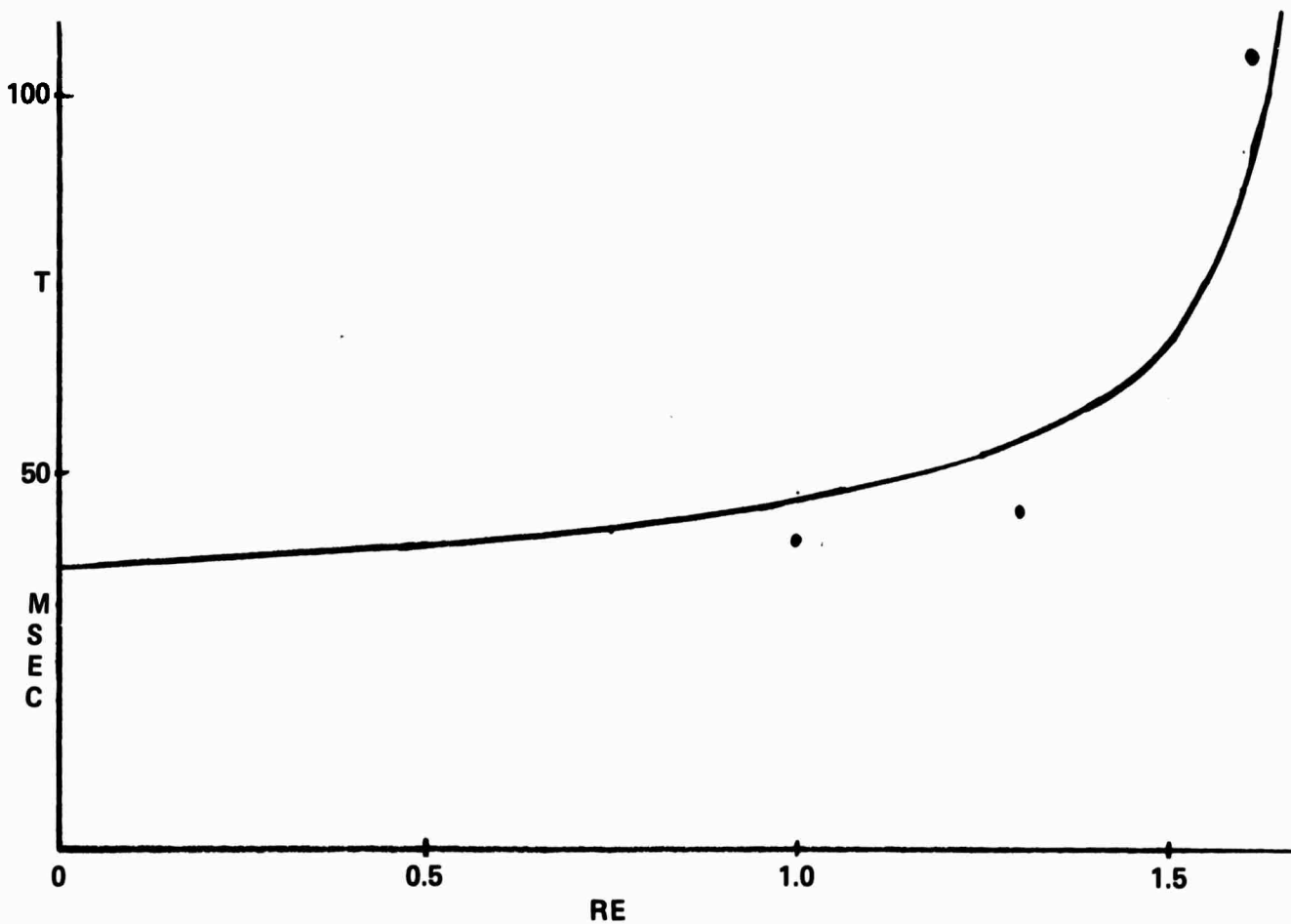
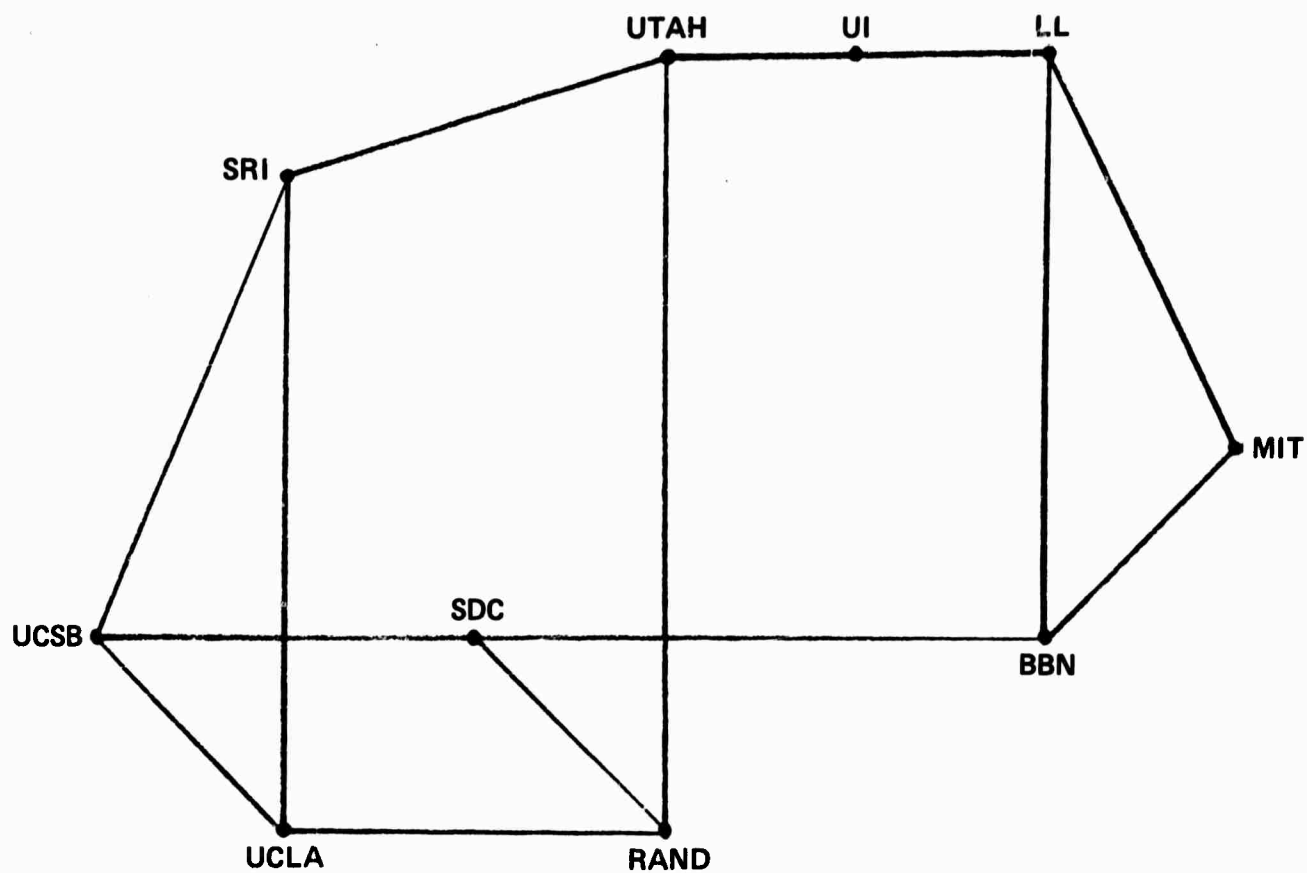


Figure 3. 10 Node Net—Configuration B

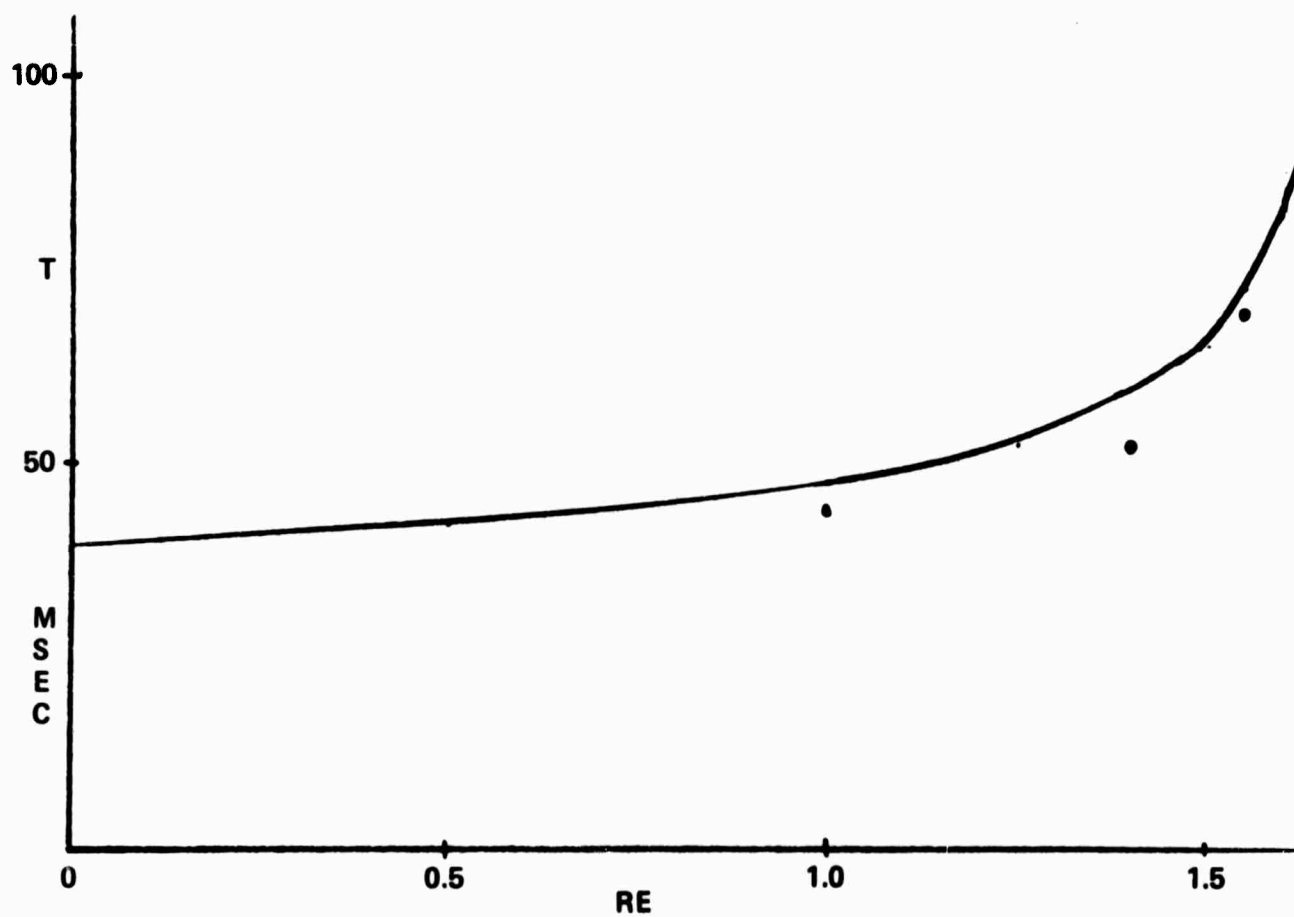
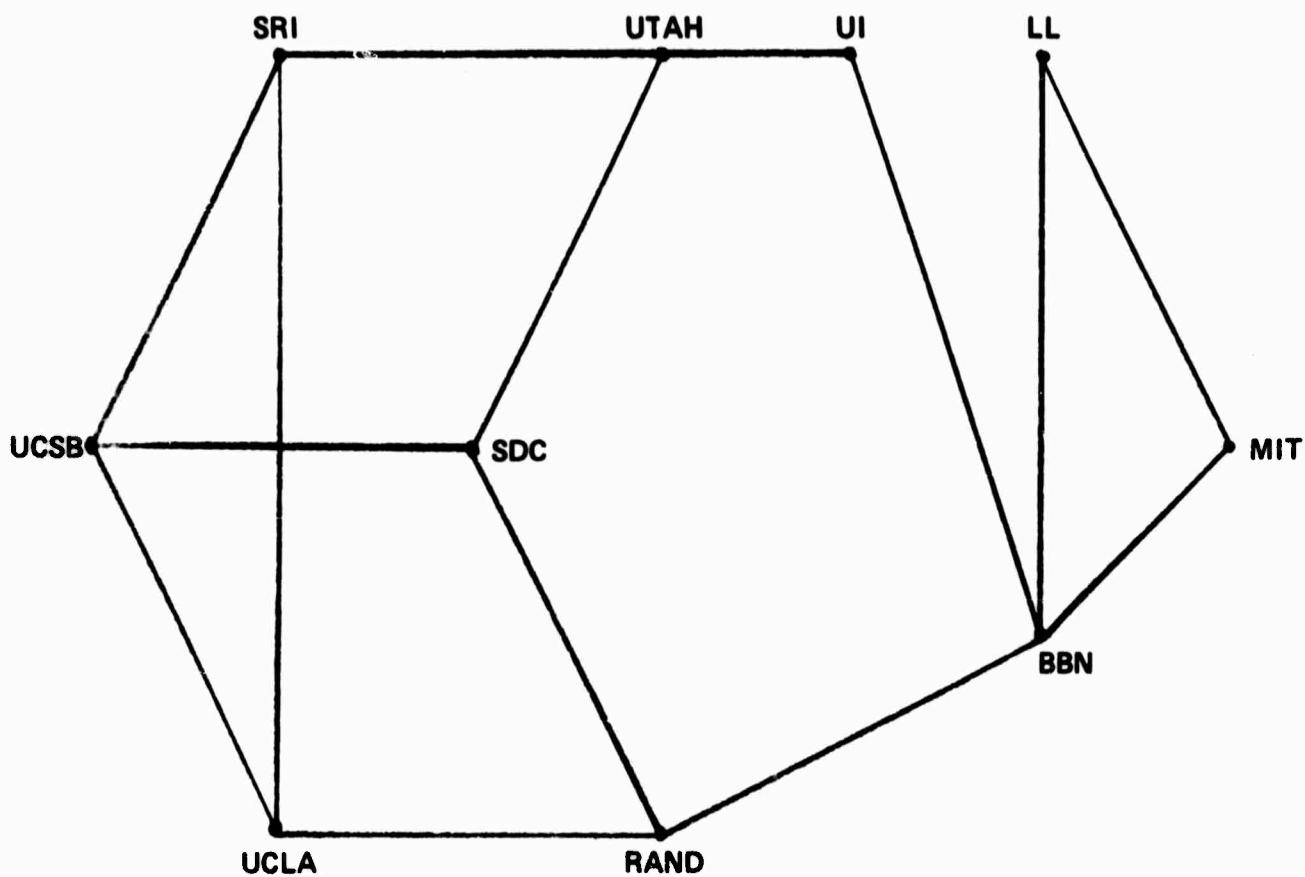


Figure 4. 10 Node Net - Configuration C

2. Long Message Model

In the long message model, messages can be two through eight packets in length. In addition, files can be composed of many messages. Here we wish to calculate the average message delay. For file structures, the average message delay is calculated assuming all messages contained in the file are created at the same time. Message reassembly time must also be considered.

The model, denoted as M1, is given by

$$T_p = \sum \frac{R_1}{R_T} \left[\frac{\mu_P}{C} + W_{P_1}(R_1, R'_1) + TL_1 + T_{cpu} \right] + T_{cpu}$$

$$T_M = \sum \frac{R_1}{R_T} \left[\frac{\mu_M}{C} + W_{M_1}(R_1, R'_1) + TL_1 + T_{cpu} \right] + T_{cpu} + \overline{RAT}$$

T_p = Average message delay for single packet messages

T_M = Average message delay for long messages and files

μ_M = length of a full packet in bits

$$W_{P_1}(R_1, R'_1) = \frac{\frac{1}{2} \left[\rho_{A_1} \frac{\mu_A}{C} + \rho_{P_1} \frac{\mu_P}{C} (1 + C_P^2) + \rho_{M_1} (1 + C_{M_1}^2) \frac{\mu_M}{C} \right]}{(1 - \rho_{A_1}) (1 - \rho_{A_1} - \rho_{P_1})}$$

$$\rho_{M_1} = \frac{(1 - M) R_1}{C}$$

$$\rho_{P_1} = \frac{M R_1}{C} \quad \rho_A = \left[\frac{M \mu_A}{\mu_P} + (1 - M) \frac{\mu_A}{\mu_M} \frac{R'_1}{C} \right]$$

$$W_{M_1}(R_1, R'_1) = \frac{\frac{1}{2} \left[\rho_{A_1} \frac{\mu_A}{C} + \rho_{P_1} \frac{\mu_P}{C} (1 + C_P^2) + \rho_{M_1} (1 + C_{M_1}^2) \frac{\mu_M}{C} \right]}{(1 - \rho_{A_1} - \rho_{P_1}) (1 - \rho_{A_1} - \rho_{P_1} - \rho_{M_1})}$$

$$C_{M_1}^2 = \sum_j P_{I_j} \left[\frac{\mu_{I_j}}{\mu_M} (1 + C_{I_j}^2) - 1 \right] \frac{R_I}{R_1}$$

R_I = Node I input data rate from its host

P_{I_j} = % of R_I destined to node j which flows on line 1

μ_{I_j} = E[message length] destined to node j from node I on line 1

$C_{I_j}^2 = \frac{\sigma_{I_j}^2}{\mu_{I_j}^2}$ = Coefficient of variation

RAT = Average message reassembly time

In addition to this model, we have considered a second model, denoted as M2, which will not be discussed in detail. Only the computational results of M2 will be presented. Table 1 shows the theoretical and experimental results for the long message case and Table 2 shows the results for the file transmission case.

From the tables, we see that T_p is reasonably predicted, but the theoretical predictions for T_M are generally too small. Improved models to predict T_M are currently being developed.

B. Node Storage Sizing

Let us now consider the average storage \bar{S}_I required per unit time for each node I in the net and the $VAR(S_I)$.

For each packet flowing through a node, we determine the amount of time T_S the packet requires storage (assuming L bits in the packet). For packets arriving at a rate λ_j per sec. ,

TABLE 1
LONG MESSAGE TRANSMISSION MODEL

RE	Model	<u>M = .25</u>		<u>M = 0.0</u>
		<u>T_P(msec)</u>	<u>T_M(msec)</u>	<u>T_M(msec)</u>
1.0	M2	48.2	158.94	188.4
	Simulation	50.9	270.19	258.2
	M1	56.6	178.9	188.4

TABLE 2
FILE TRANSMISSION MODEL

RE	Model	<u>M = .85</u>		<u>M = .75</u>		<u>M = 0</u>
		<u>T_P(msec)</u>	<u>T_M(sec)</u>	<u>T_P(msec)</u>	<u>T_M(sec)</u>	<u>T_M(msec)</u>
.75	M2	47.3	445.2			
	Simulation	47.8	419.2			
	M1	60.3	466.8			
.9	M2	49.6	451.9			
	Simulation	50.6	552.6			
	M1	67.5	488.4			
1.0	M2	51.6	459.9	55.1	467.9	615.8
	Simulation	60.6	806.5	61.7	790.3	807.1
	M1	73.8	514.7	84.7	537.8	615.8

$$S_I = \sum_j \lambda_j T_{S_j} L_j .$$

T_{S_j} , with its associated λ_j and L_j must be calculated for three separate cases: (a) packets arriving on node input lines, (b) packets exiting on node output lines, and (c) packets being transferred between the node and its associated HOST.

Theoretical equations have been derived for \bar{S}_I for the single packet message case and some preliminary hand computations have shown that the theoretical model and simulation results agree very well. When the analysis for σ_{S_I} is completed, these equations will be programmed and computed as a function of the effective net data rate RE.

We must extend the model to include the effect of long messages.

C. For the rest of the performance messages listed in Section I, only a small amount of effort has been expended to date on their calculation. Therefore, no definitive results are worth mentioning at this time.

III. Message Routing Techniques

We have classified message routing techniques into three categories:

1. Fixed routing techniques.
2. Local Routing techniques.
 - a) A node only has information about network congestion and delay from messages passing through it.
 - b) A node has the same information as (a) above plus its queue lengths and has access to its nearest neighbor node delay tables.
3. Global routing techniques.
 - a) Network routing control center (NRCC). The NRCC calculates routing tables for each node in the network.
 - b) Ideal observer routing. Knowing the events which will occur in the future, calculate the message routing to minimize T . This is a scheduling problem.

The analysis problems being considered are:

1. Routing algorithm specification.
2. Routing algorithm performance measures.
 - a) T (function of routing algorithm).
 - b) Stability and convergence of the algorithm.
 - c) Looping phenomenon (packets trapped in loops).
 - d) Adaptability of the algorithm to network changes.
3. Interrelationship between specific routing algorithms and the network topology.

Rand has extensively investigated the local, Type A, routing algorithms. To date, we have investigated the fixed routing algorithms and the local, Type B, routing algorithms. Below, we discuss our results in more detail.

Fixed Routing Techniques

1. Evaluation of average message delay. The results of this investigation were presented in Section II.
2. Fixed routing matrix solution technique.

Problem

$$\text{Minimize}_{\{R.M.\}} T(RE) = \sum_{i=1}^N \frac{\lambda_i}{\gamma} T_i \quad (3)$$

subject to link capacity constraints $C_1 = C$. R.M. is the fixed routing matrix for the network. This problem is essentially a multicommodity flow problem with non-linear cost constraints.

The computational technique for this algorithm will not be given here, but is rather simple and requires few iterations. Figure 5 shows the results of the algorithm. Each curve represents T versus RE for the R.M. that minimized Eq. (3) at a specific value of RE . The lower envelope of all the curves then represents the solution to Eq. (3) for all RE . The only theoretical solution point we know is T_{sp} , the solution to any of the shortest path algorithms for $RE = 0.0$.

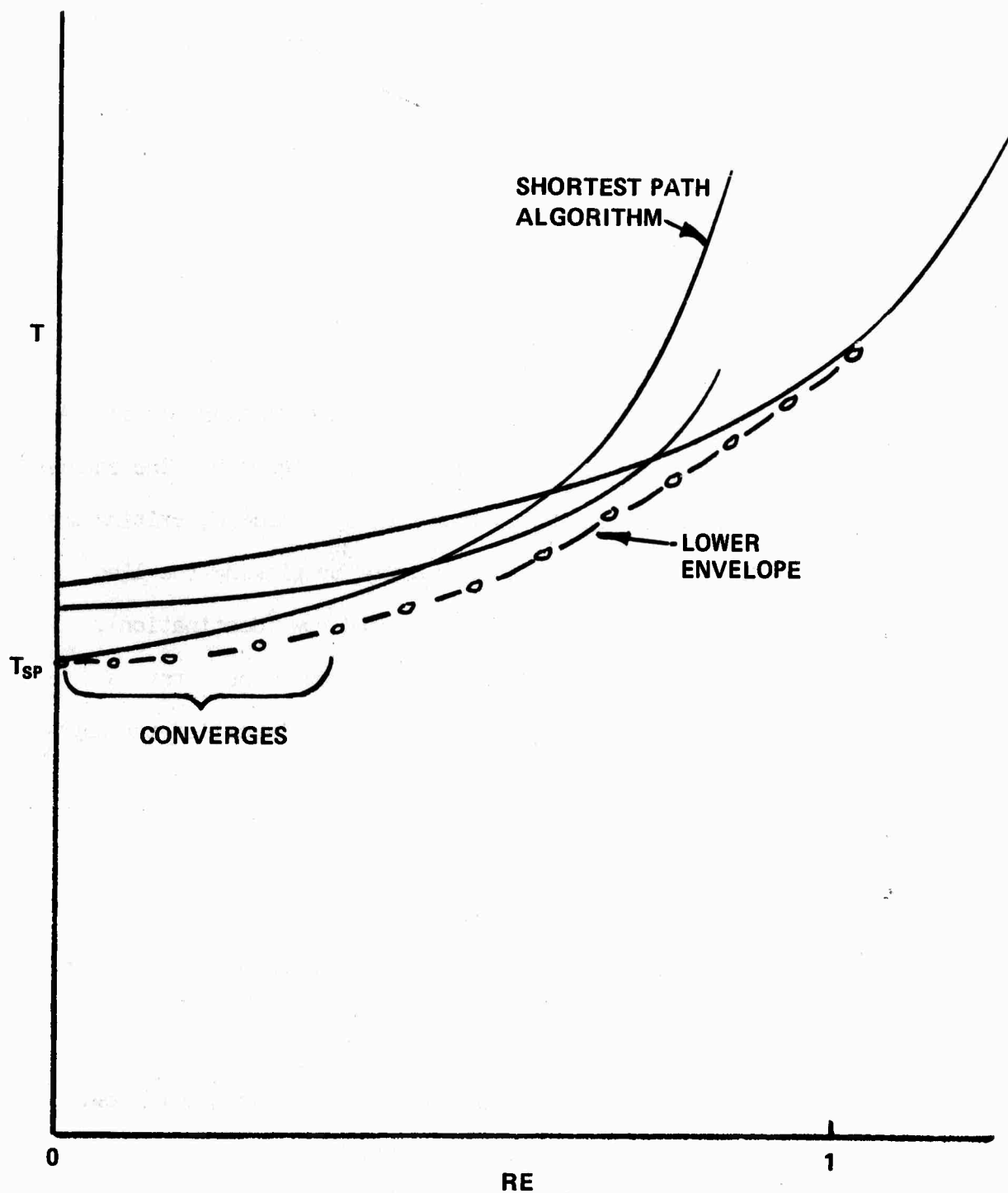


Figure 5. Minimization of $T(RE)$

Comments on the operation of the algorithm follow. Let T_K be the solution to Eq. (3) for the K^{th} iteration of the algorithm.

1. For small RE, the algorithm converges ($T_{K+1} = T_K$).
2. For medium RE, the T_K 's oscillate in some random manner.

By letting K be large, we can find $T = \text{Min} (T_0, T_1, \dots, T_K)$.

3. For large RE, the algorithm wants more line capacity than is available.

We intend to modify the algorithm to improve its operation.

Local, Type B, Routing Techniques

All of the local routing algorithms operate in a similar manner. A delay table is constructed in each node as shown in Figure 6. The entries $\hat{T}_j(N, K)$ are the estimated delays to go from node j to node N , exiting the node on line K . The best route would be selected by picking the line number whose row entry is minimum for the selected row (destination).

We have considered two types of local algorithms. The first is called the periodic updating algorithm and the second is called the asynchronous algorithm.

Periodic Updating Algorithm Operation

1. Routing is held fixed for K Msec.
2. Then modify the delay tables by the change in the line queue lengths during the K Msec. period.
3. Form a vector \underline{M} which contains the minimum \hat{T} 's from each row.
4. Transmit the \underline{M} to all nearest neighbors.

		NODE OUTPUT LINE NUMBER		
		1	2	K
DESTINATION NODE	1			
	2			
		•	•	•
		•	•	•
		•	•	•
	N			$\hat{T}_J(N, K)$

Figure 6. Delay Table in Node J

5. When a node receiving the \underline{M} 's from its neighbors, it adds its current queue length plus a delay factor D_p to all entries and places it in the proper column of its delay table.
6. After all \underline{M} 's are received, find the line number for each destination which corresponds to the minimum delay estimate for a row.
7. Hold this fixed route selection for the next K Msec.

This algorithm suffers from looping and is only stable for $D_p > a$ couple of $E[\text{packet lengths}]$ for $K = 500$ Msec. The parameter D_p controls the degree of alternate routing.

Asynchronous Updating Routing Algorithm Operation

1. Minimum delay route picked on a per packet basis.
2. Delay tables are constantly updated by queue length changes.
3. If delay table entries change by more than a preset threshold, the vector \underline{M} is formed, and the node's nearest neighbors are informed of the change.
4. If a node receives a vector \underline{M} , the delay table is updated as in Step 5 of the periodic updating algorithm.

The algorithm's stability is a function of D_p and the threshold setting.

Figure 7 shows the performance of the adaptive routing techniques for the 19-node net shown in Figure 1 and for $RE = 1.0$. An attempt was

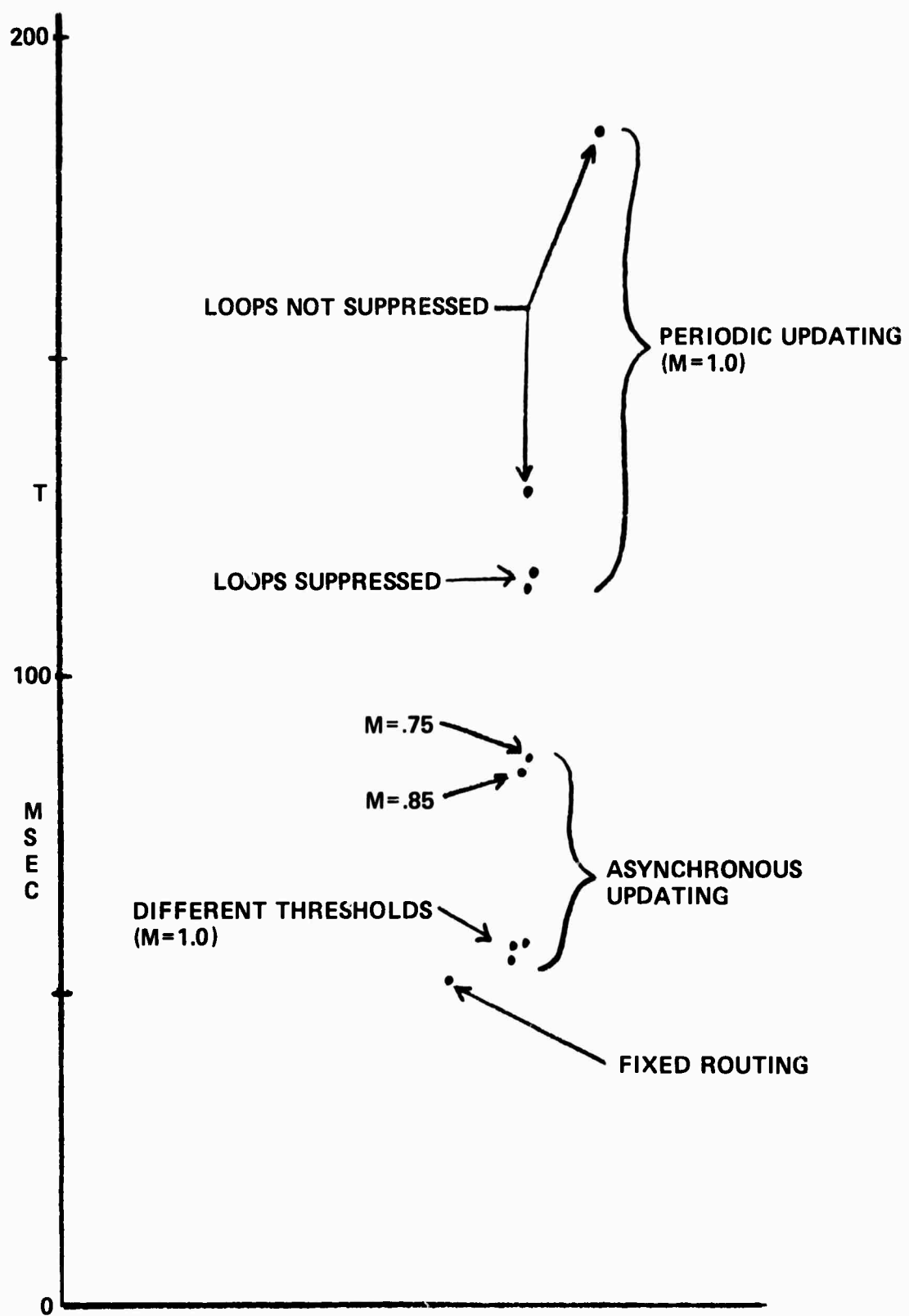


Figure 7. Local, Type B, Routing Algorithm Performance

made to suppress looping in the periodic updating algorithm, but did not significantly improve the performance of the algorithm. We are continuing the investigation of this class of routing algorithms.

IV. Message Flow Control and Acknowledgment Procedures

Strategies must be specified to aid in the control of message flow through the network. These strategies will be coupled into the routing doctrine and models. Some of the facets which are being considered are:

1. Node input choking.
2. Message priorities.
3. Links down.
4. Nodes down.
5. Node storage not available for relay traffic.
6. Information to be carried by acknowledgment messages.
7. Classes of messages used only for network control functions.

APPENDIX G

DYNAMIC STORAGE ALLOCATION FOR BINARY SEARCH TREES
IN A TWO-LEVEL MEMORY

by

Richard Muntz and Robert Uzgalis

BLANK PAGE

DYNAMIC STORAGE ALLOCATION FOR BINARY SEARCH TREES IN A TWO-LEVEL MEMORY

Richard Muntz and Robert Uzgalis
Department of Computer Science,
School of Engineering and Applied Science,
University of California, Los Angeles.

Summary

Binary search trees have long been recognized as useful structures for storing and retrieving data. When the tree is too large to be stored in main memory, the cost of accessing portions of the tree in secondary storage becomes a critical problem. It is assumed here that transmissions from secondary storage to main memory are accomplished in fixed size units called pages. In this paper we propose an heuristic for allocating storage to the nodes of the search tree so as to minimize the average number of page requests necessary to search the tree. This method of allocation is compared with a sequential allocation algorithm. Experimental and analytic results are presented.

Introduction

In the binary search trees we consider, each node contains a data item. In many applications the data items to be stored are not all available initially but are received in some arbitrary order. It is this latter case we assume in this paper. It is clear that when the tree is constructed dynamically as the data is received, the structure of the tree is determined by the order in which the data arrives. An example is text analysis in which the data is the word to be stored. The purpose might be to count the number of occurrences of words in the text vocabulary. Although many common words may be placed in the tree initially (viz. 'a', 'the') most of the vocabulary will be added to the tree as it is encountered in the text.

When the search tree can be stored in main memory the most important characteristic of the tree is the average number of nodes that must be interrogated to locate a data item. However, when the tree is stored in secondary storage the average number of accesses to secondary storage becomes the critical measure of performance. It will be assumed here that secondary storage consists of a sequence of fixed sized blocks called pages. Each access to secondary storage is a request for one page to be

transmitted to main memory. The time required to search the tree will be largely a function of the number of distinct pages that are referenced in a search path, i.e. the number of page transitions. The number of pages referenced is determined partially by the method of allocating storage to nodes. Since the tree grows dynamically the method of allocation must be dynamic.

Description of Allocation Techniques

We now describe two methods of allocating storage to nodes as they are added to the tree. The first method, Sequential Allocation, ignores page boundaries and allocates storage for new nodes in consecutive locations as they are received. This is certainly the most straight forward allocation scheme. Note however that the nodes on a given page are related by locality in the input sequence and not necessarily by locality in the tree.

The second method, Grouped Allocation, explicitly takes page boundaries into consideration. When adding a node to the tree we first locate the node to which the new node will be linked. This node will be the father of the new node in the search tree. If the father node is on a partially filled page then the new node is allocated space on this page. If the father is on a full page then a new page is allocated to the tree and the new node is allocated at the beginning of this page. The first node allocated on a page is called the seed node for that page. Note that, only nodes in the subtree of the seed node will be allocated to this page. It is clear from this description that nodes on the same page will be related by locality in the search tree and intuitively we expect this scheme to result in fewer page transitions.

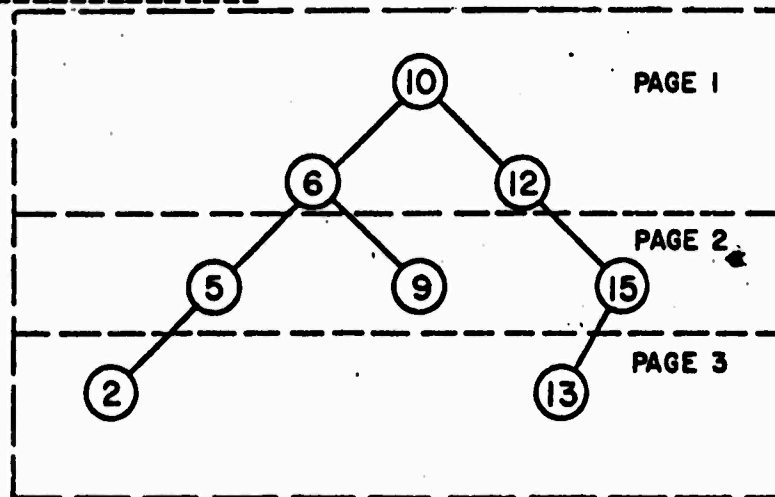
An example is given below to illustrate the two allocation schemes just described.

Example

Page size = 3 nodes

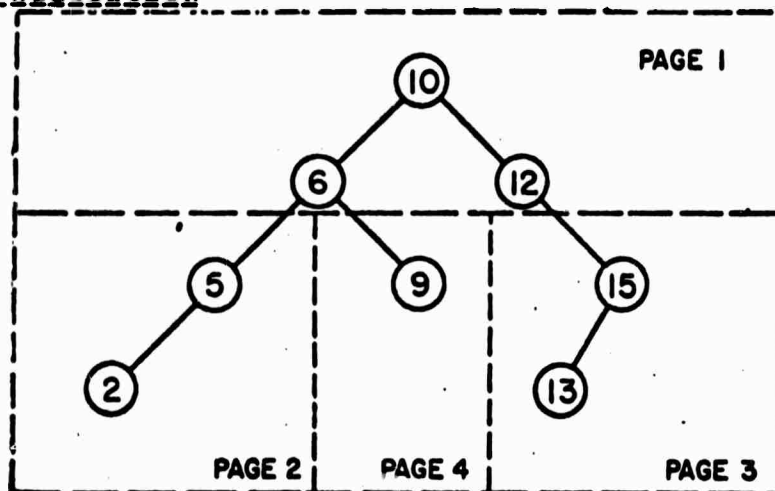
Input sequence 10, 6, 12, 5, 15, 9, 2, 13.

Sequential Allocation



The dotted areas indicate nodes on the same page. If all nodes are accessed with equal probability the average number of page transitions in a search is $15/8$.

Grouped Allocation



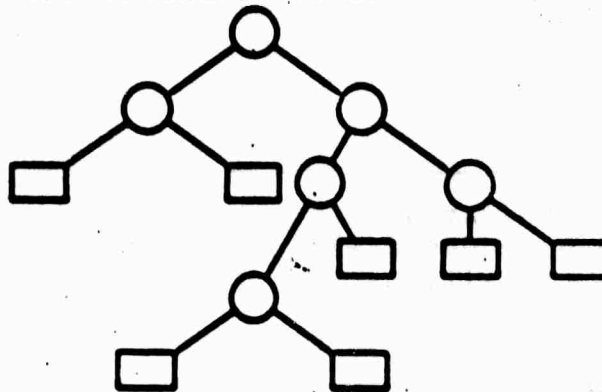
Average number of page transitions in a search is $13/8$.

The grouped allocation scheme will in general result in extra pages allocated for storage of the tree. However, a simple modification can be made to the algorithm which effectively eliminates this disadvantage. Since this modification would complicate the comparison of the methods unnecessarily, we will first discuss results for the two schemes as described. The modification to the grouped allocation technique will be discussed later.

Comparison of Allocation Techniques

Analytic Models

We will need the following definitions and results. The nodes of the binary search tree will be called internal nodes. For every internal node that has a null left (right) link we add an external node as the left (right) successor of that node. In the illustration below the circular nodes are the internal nodes and the square nodes are the external nodes.



We define the path length to a node (either internal or external) as the number of internal nodes on the path from the root of the tree to that node. The average internal (external) path length is the average path length to an internal (external) node. For example, the root itself has a path length of one. Given that a binary search tree contains N internal nodes, there are $N!$ sequences in which these nodes could have been entered in the tree. Each of these sequences we assume is equally likely. Formulas for the average internal and external path lengths, over all possible input sequences have been derived by Hibbard¹ and are given below.

For a binary search tree containing N internal nodes the average internal path length, denoted by $\bar{l}(N)$, is approximately

$$\bar{l}(N) = 1.4 \log (N) - 1.7$$

For a binary search tree containing N internal nodes the average external path length, denoted by $\bar{l}'(N)$, is approximately

$$\bar{l}'(N) = l(N) + 1 = 1.4 \log(N) - 0.7$$

Sequential Allocation

We have developed the following analysis to estimate the mean number of page transitions necessary to locate a node when the sequential allocation scheme is used. Let N be the number of nodes in the tree and let p be the number of nodes in a page. The nodes are assumed to be referenced with the same frequency.

Let r be some positive integer. After rp nodes have been entered in the tree, the range of key values has been partitioned into $rp+1$ regions. Of the next p nodes to be added to the tree, only those that fall in the same region will be linked together in the tree. For a moderate value of r we would expect very few nodes in the page to be linked together. We will use the approximation that after the first two pages each node that is interrogated on a search path requires a new page reference. This choice was made to simplify the analysis. Clearly this approximation will lead to a somewhat pessimistic estimate of the mean number of page references.

Nodes in the first page require only one page reference in their search paths. Nodes in the second page require two page references in their search paths. The nodes in the other pages require a reference to page one since the root of the tree is in page one. However, a reference to page two may not be necessary. It is shown below that for nodes outside the first two pages an average of approximately three-fourths will have search paths which pass through the second page (i.e. at least one node on the search path is in the second page). Therefore these nodes have an average of 1.75 page references to pages one and two.

A complete formal proof of this statement would be rather long so we offer the following simplified argument. Consider N boxes labeled in order with the N data keys of the search tree. Let the p nodes in the first page be represented by p black balls and the p nodes in the second page be represented by p red balls. Placing these $2p$ balls in the boxes in some random order, no more than 1 ball to a box, represents the tree after the first two pages are filled. Now consider an empty box (which represents a node that will be outside of pages one and two). We leave it to the reader to show that the path to this node will contain a node from page two if and only if the first filled box to the left or right of this box contains a red ball (i.e. the corresponding node in page two). Since there are an equal number of red and

black balls and they are distributed randomly the probability of the first filled box to the left (or right) containing a red ball is approximately $1/2$. Therefore the probability of a red ball in the first filled box in either direction is approximately $3/4$.

For that portion of the search path that is outside pages one and two, we assume that each node requires a page reference. Therefore we need to calculate the mean path length for nodes outside the first two pages and then subtract the mean length of the portion of the search path in pages one or two.

Let the mean path length of nodes outside of pages one and two be denoted by L . Then

$$\bar{x}(n) = \frac{2p}{N} \bar{x}(2p) + \frac{N-2p}{N} L$$

$$L = \frac{N}{N-2p} [\bar{x}(N) - \frac{2p}{N} \bar{x}(2p)]$$

The mean length that must be subtracted from L is just the mean external path length for a tree with $2p$ nodes, i.e. $\bar{x}'(2p)$. Therefore the average number of page references required by nodes outside the first two pages, denoted by α , is

$$\alpha = 1.75 + L - \bar{x}'(2p)$$

$$\alpha = 1.75 + \frac{N}{N-2p} \bar{x}(N) - \frac{2p}{N-2p} \bar{x}(2p) - \bar{x}'(2p)$$

$$\text{using } \bar{x}'(2p) = \bar{x}(2p) + 1 \text{ we get}$$

$$\alpha = 0.75 + \frac{N}{N-2p} [\bar{x}(N) - \bar{x}(2p)]$$

Since we also know the number of page references required by nodes in pages one and two, we can compute the average number of page references for the entire tree (denoted $\bar{S}(N,p)$).

$$\bar{S}(N,p) = \frac{p+2p+(N+2p)[0.75 + \frac{N}{N-2p} (\bar{x}(N) - \bar{x}(2p))]}{N}$$

$$\bar{S}(N,p) = \frac{3p}{N} + \frac{0.75 - 1.5p}{N} + \bar{x}(N) - \bar{x}(2p)$$

$$\bar{S}(N,p) = 0.75 + \frac{1.5p}{N} + 1.4 \log_2 \left(\frac{N}{2p} \right)$$

Grouped Allocation

In the grouped allocation technique each page contains the top portion of a subtree in the search tree. The average internal

path length in the search tree can be approximated by $1.4 \log_2(N) - 1.7$. The average external path length within a page is similarly approximated by $1.4 \log_2(p) - 0.7$. Therefore a crude estimate of the average number of page transitions in a search path (denoted $\bar{G}(N, p)$) is

$$\bar{G}(N, p) = \frac{1.4 \log_2(N) - 1.7}{1.4 \log_2(p) - 0.7}$$

These two formulas and experimental results for $N = 2048$ and various page sizes are illustrated in figure 1.

Modified Group Allocation

As mentioned previously the grouped allocation scheme can result in many sparsely filled pages allocated to the search tree. If the total size of the search tree is known or can be estimated the grouped allocation scheme can be modified to limit the number of pages allocated. The method is simply to allocate new pages to seed nodes until there are no more pages available. After this, seed nodes are allocated to partially full pages. The selection of a page in which to plant the seed node is selected so as to distribute them evenly in the partially full pages.

In figure 2 we illustrate the results of an experiment in which the number of pages available was varied from the minimum (i.e. $\lceil N/p \rceil$) to an unlimited number of pages. Use of the minimum number of pages to store the tree resulted in a slight increase in page transitions. This seems to indicate that the beneficial effects the method are due to the early stages of allocation. The lowest curve in figure 2 is the minimum average number of page references possible. The formula for this lower bound is discussed below.

Let N be the number of nodes in the search tree. Then there are at least $\lceil N/p \rceil$ pages in which the tree is stored. Now consider a page tree in which each node corresponds to a page. there are p nodes of the original tree on each page and thus the tree will be a $(p+1)$ -ary tree. The average path length in this page tree is the average number of page transitions required per search. The average path length in the page tree will be minimized if it is balanced. A slight variant of a formula from Knuth² gives the average path length in a balanced $(p+1)$ -ary tree of $n = \lceil N/p \rceil$ nodes as

$$\frac{1}{n} \left[\left(n + \frac{1}{p} \right) q - \frac{((p+1)^{q+1} - (p+1))}{p^2} \right] + 1$$

where

$$q = \lceil \log_{p+1} (pn + 1) \rceil$$

$$n = \lceil \frac{N}{p} \rceil$$

It is interesting that our experimental data indicates that the grouped allocation scheme results in only 10 to 20% more page references than this lower bound. This is particularly interesting since part of the difference is due to the fact that the search tree is not balanced in general.

Further Reduction of Page References

It is clear that if all of the search tree is in secondary storage the average number of page references must be greater than or equal to one. From the formula for $G(N,p)$, we see that for $p = \sqrt{N}$ the mean number of page references is approximately 2. But one of these page references can be eliminated if the first page (i.e. containing the peak of the search tree) is retained permanently in memory. A buffer of size p is also necessary to temporarily store pages brought in from secondary storage. Therefore, with a dedicated portion of main memory of size $2p$ (or $2\sqrt{N}$) the average number of page faults will be close to one. For example, if $N = 40,000$ nodes and $p = 200$ only 1% of the tree or 400 nodes need be stored in main memory to achieve a near minimum of page references.

Conclusion

We have described two methods of dynamic storage allocation for trees in a paged environment. The grouped allocation technique which we developed was shown to reduce the average number of page references per search over a wide range of page sizes. With a slight modification to the scheme no extra storage is necessary. With a large search tree it is possible to reduce the number of references to secondary storage to close to one by keeping one or two percent of the tree in main memory.

Acknowledgments

This research was supported by the National Science Foundation (GJ-809) and the Department of Defense Advanced Research Projects Agency (DAHC15-69-C-0285). Computing support was provided by UCLA Campus Computing Network. Reproduction in whole or in part is permitted for any purpose of the United States Government.

References

1. Hibbard, T. N., "Some Combinatorial Properties of Certain Trees with Applications to Searching and Sorting", J.ACM 9, 1(Sept. 1962) pp. 13-28.
2. Knuth, D. E., The Art of Computer Programming, Vol. I, Addison-Wesley, 1968, p.401.

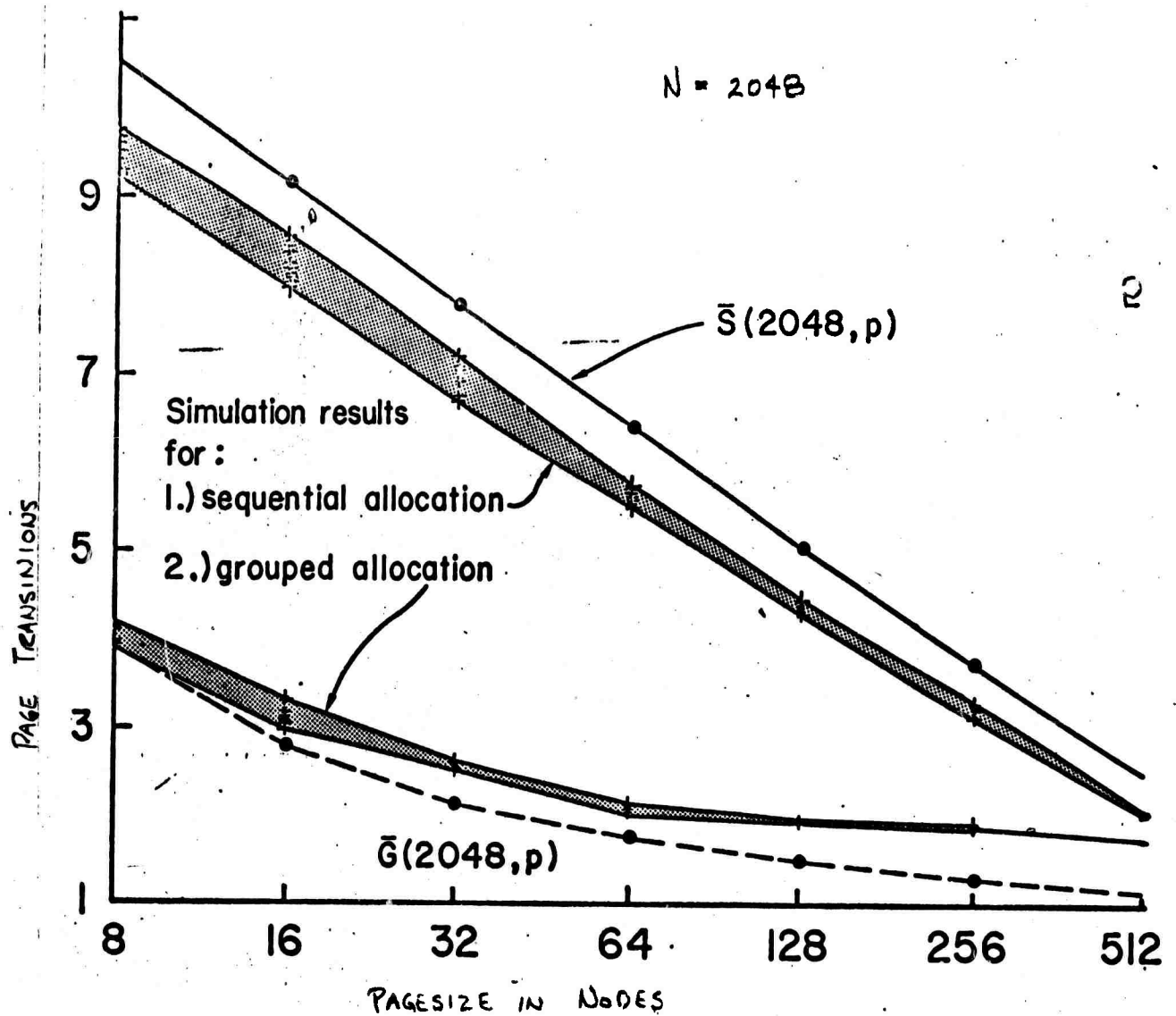


Figure 1. Comparison of Sequential and Grouped Allocation

$N = 2048$

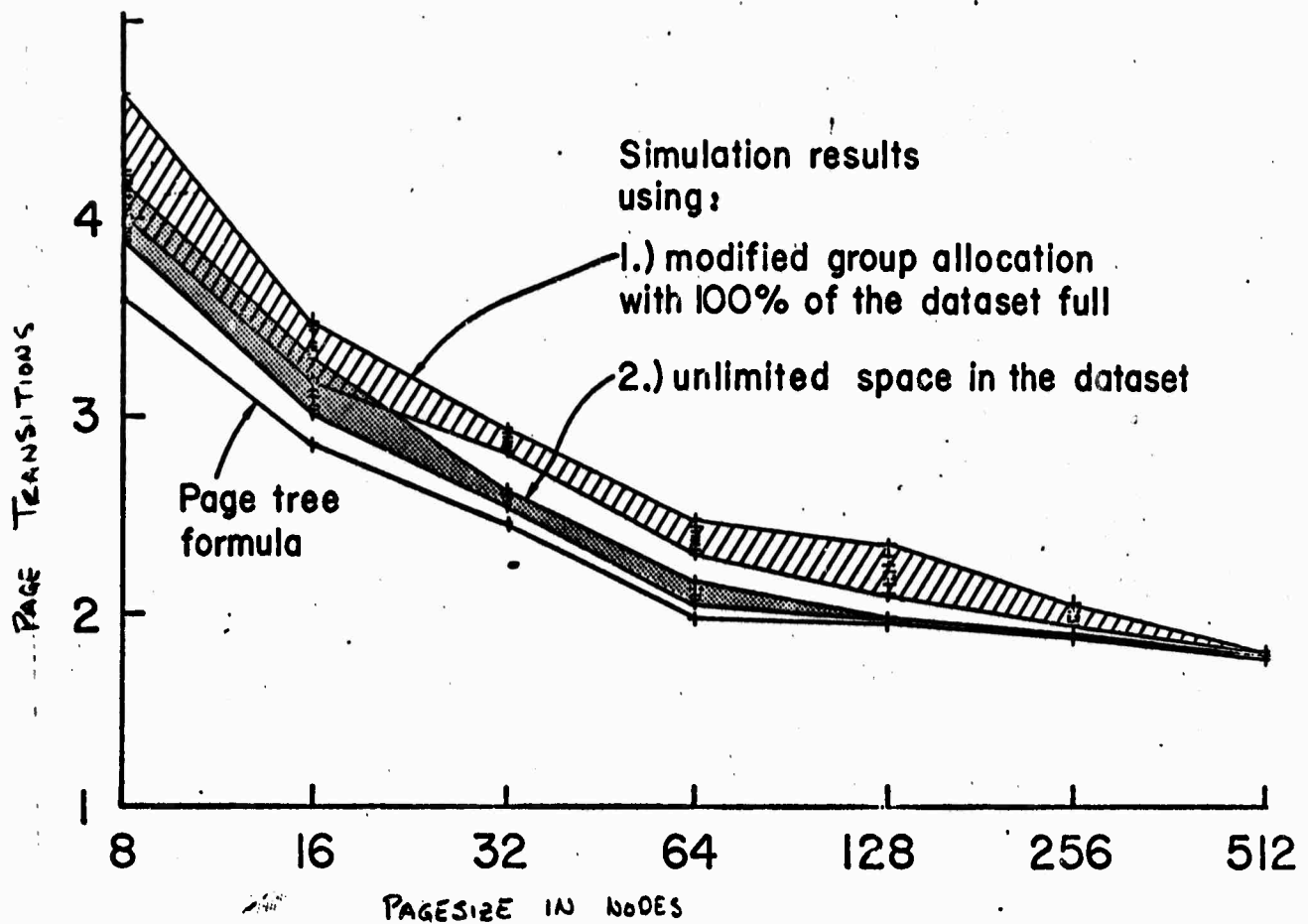


Figure 2. Simulation Results for the Grouped Allocation Techniques

2. Simulation Results for the Grouped Allocation Techniques

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) School of Engineering and Applied Science Computer Science Department 405 Hilgard, University of California, Los Angeles		2a. REPORT SECURITY CLASSIFICATION Unclassified	
		2b. GROUP	
3. REPORT TITLE Computer Network Research			
4. DESCRIPTIVE NOTES (Type of report and inclusive dates)			
5. AUTHOR(S) (First name, middle initial, last name) Leonard Kleinrock			
6. REPORT DATE August 15, 1970		7a. TOTAL NO. OF PAGES 109	7b. NO. OF REFS 26
8a. CONTRACT OR GRANT NO. DAHC-15-69-C-0285		9a. ORIGINATOR'S REPORT NUMBER(S)	
b. PROJECT NO.			
c.		9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
d.			
10. DISTRIBUTION STATEMENT Distribution of this document is unlimited.			
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY	
13. ABSTRACT ARPA Semiannual Technical Report, February 15, 1970, to August 15, 1970			